# Vidispine REST API Documentation

*Release 5.x*

**Vidispine AB**

**Nov 23, 2021**

CONTENTS


1 Introduction and data model 3

2 Items and Metadata 13

3 Collections and Libraries 93

4 Shapes, Components and Transcoding 103

5 Storages and Files 139

6 Jobs and Task Definitions 167

7 Notifications 181

8 Resources 185

9 Timelines and sequences 215

10 Users, Groups, and Access control 231

11 Multi-site 249

12 Miscellaneous Topics 251

13 Monitoring 257

14 Configuration and Integration 265

15 Troubleshooting and obtaining information 327

16 Installation 331

17 API Reference 345

18 Release Notes 913

HTTP Routing Table 1045

Index 1055

The Vidispine REST API is a rich interface for creating custom media management solutions for the most complex requirements.

This documentation is available as PDF here. The documentation comes with its own searching functionality, in the upper left corner.

This reference documentation is divided into the following sections. Each section starts with an overview and is then followed by introductory guides. The API reference section at the end explains the API and resources in detail.

CHAPTER

ONE

# INTRODUCTION AND DATA MODEL

## 1.1 Entities in Vidispine

Before start playing with the API, a short introduction to the data model might be valuable. The figure *Overview of the entities in Vidispine* shows some of the entities that builds the assets in Vidispine.

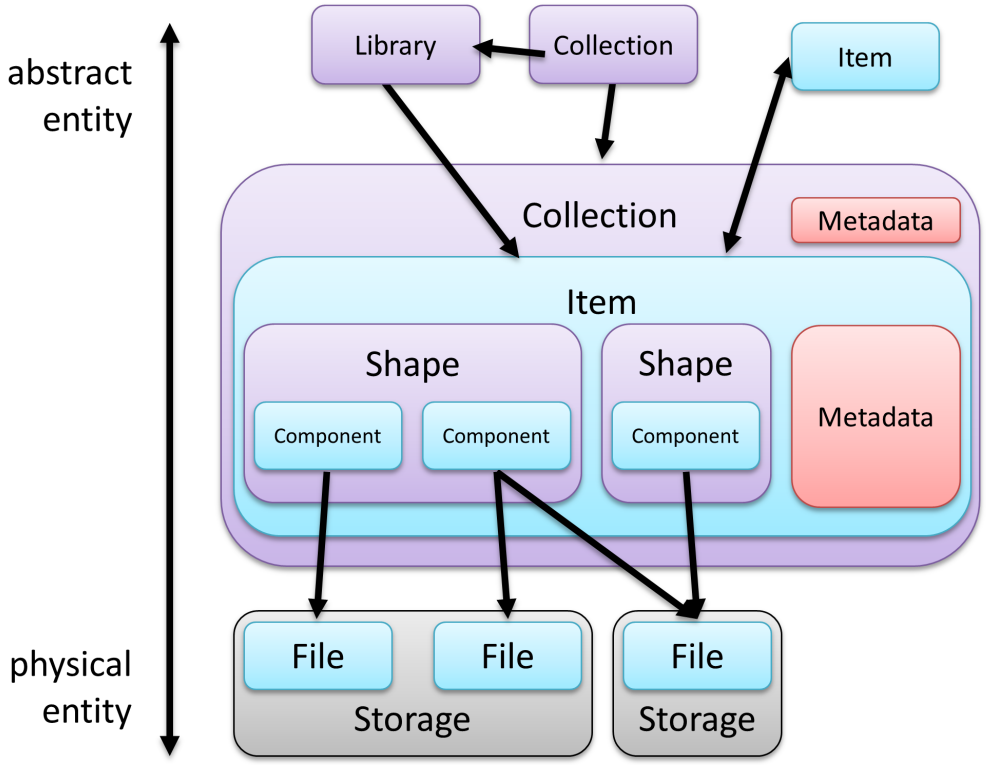


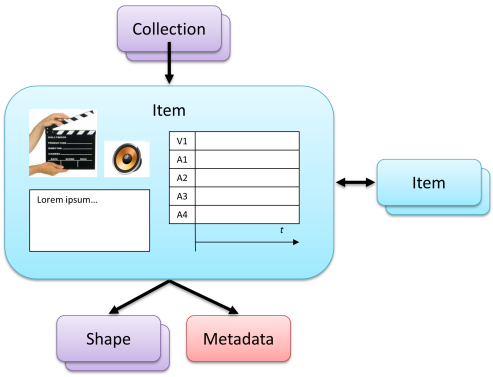

Fig. 1.1: Overview of the entities in Vidispine

### 1.1.1 Item

The item is the central piece in the data model. This corresponds to an *asset* in other systems. The item is an abstraction of the physical content (*essence*) and holds information about the content (*metadata*). For information about how to create items, see *Imports*.

Other entities, further down in the hierarchy, may also hold metadata. The item has the richest functions for how metadata can be stored, searched, and indexed. For information about metadata on an item, see *Item metadata*.

The item also holds information about which users that are allowed to read and modify information (*access control*). For information about access control, see *Access control for items, libraries, collections*.



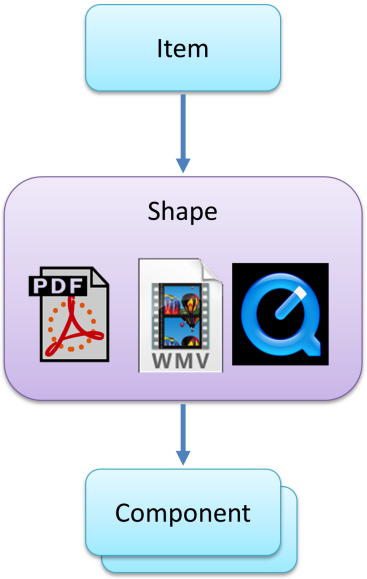

### 1.1.2 Shape

A shape is a physical rendition of an item.

- For a video, it can be a low-resolution editing version, a web version, ...
- For a document, it can the the pages as images, extracted text, ...
- Etc.

For information about shapes, see *Item shapes*.

A shape can have one or several shape tags. The shape tags are used when Vidispine selects which files that are being transcoder, exported, thumbnailed, etc. A special shape tag is `original`, a shape tag that the imported source file gets. The shape tag also contains the recipe for how to create new shapes using the transcoder. For information about shape tags, see *Shape tags and presets*.




### 1.1.3 Component

Each shape has one or more components. A media shape might for example contain:

- A container component
- Video components
- Audio components

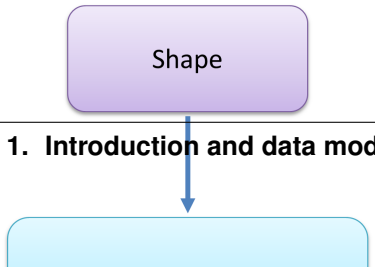
Each component corresponds to one file content. There may be several copies of the file however.

The component contains information (*technical metadata*) about codes, resolution, frame rate and more. For information about components, see *Item shapes*.

### 1.1.4 File and storage

The file entity represents a physical file on a file system. The file is stored on a storage. Vidispine manages all files, and knows which copies of a file that have been made, and how they relate.

For information about files and storages, see *Storages*.



### 1.1.5 Library

A library is a list of items. A library can be created manually, by adding the items to a library, or dynamically, by adding search results to a library. Libraries are useful when performing batch operations. Libraries can also be used when creating *rules*.

For information about libraries, see *Libraries*.

### 1.1.6 Collection

A collection is a list of items, libraries, and/or collections. Collections may have metadata and access rights, which are applied to the items that belong to the collection. While the library is typically created from a search operation, the collection is often used like a file system folder, to organize items.

For information about collections, see *Collections*.

## 1.2 RESTful API

The Vidispine API is a REST API (http://en.wikipedia.org/wiki/Representational_State_Transfer), using HTTP as a transfer protocol.

### 1.2.1 Some basics in the RESTful API

#### URI

The URI is used as a resource (noun). This means each entity in Vidispine has its own (base) URI. Example:

- `/API/item` - All items
- `/API/item/VX-204` - A particular item
- `/API/item/VX-204/shape` - All shapes for a particular item
- `/API/item/VX-204/shape/VX-576` - A particular shape for a particular item

#### Method

The HTTP method is used as a verb. The verb is used to specify whether to Create (POST), Read (GET), Update (PUT) or Delete (DELETE) an entity. This is called CRUD (http://en.wikipedia.org/wiki/Create,_read,_update_and_delete).

GET

- Get list of items/jobs or storage definition etc.
- Does not change anything to the database.

POST

- Start jobs, create new collection, etc.
- Will create one or more new entities in the database.

PUT

- Update existing entity, create new entity with supplied id.
- Identical sequential requests will not create new entities.

DELETE

- Delete items, abort jobs, etc.
- Identical sequential requests will not change anything (fails gracefully on subsequent requests).

#### Media type

Media types are important. To specify which media type the *request* has, HTTP header `Content-type` is used. To specify which media type the caller accepts as response, HTTP header `Accept` is used. Most methods in Vidispine read XML (`application/xml`) or JSON (`application/json`) and write XML or JSON. Some methods reads and/or writes text (`text/plain`), though.

#### Parameters

Parameters are given as *query* parameters or *header* parameters.

#### Query parameters

Given at the end of the URI. The query parameters follows after a question mark (`?`), and each query parameter key/value pair is delimited by an ampersand (`&`). An equal sign = is used to separate key and value. Keys and values have to be URL encoded.

#### Header parameters

Header parameters are given in addition to the URI. The `Content-type` and `Accept` headers have already been mentioned. Other header worth mentioning is the `RunAs` header used for authentication (*Run-As option*), and the `index` and `size` header, used at import (*Import using the request body*).

## 1.3 Common elements in the API

### 1.3.1 Identifiers

Most entities in Vidispine are identified by a `Site ID`. A Site ID is a string of the form: { *site* } – { *serial* } (example: `ATL-3033` ). Note that a Site ID is not unique within the system, there could be both an item and a job with the Site ID `VX-195`, thus Site IDs are only unique within the entity type.

See also *External identifiers*.

---

**Note:**

- **site** is by the following regular expression form: `[_A-Za-z][_A-Za-z0-9]*`. The default site name is `VX`. This can be overridden with the Java system property `com.vidispine.site`.
- **serial** is of the following regular expression form: `[1-9][0-9]*`
- **site** is maximum 10 characters, and case sensitive

---

#### Long identifiers

In order to avoid confusion with non unique identifiers, it is possible to have Site IDs displayed as `ITEM-VX-1`, `JOB-VX-1`, `STORAGE-VX-3`, etc. To do this, add the *Java system property* `vidispine.identifier.format` with the value `full`. After this is done, a *re-index* of items and collections should be started. Now identifiers presented in the system will be of the form described above.

### 1.3.2 Boolean operators

XML elements to handle boolean expressions:

#### or

```
<or>
  <matching-expression />
  ...
</or>
```

#### and

```
<and>
  <matching-expression />
  ...
</and>
```

#### not

```
<not>
  <matching-expression />
</not>
```

### 1.3.3 Text/plain formatting

#### CR LF

CRLF is used in *text/plain* representation when several values are returned, such as tuples or lists. CRLF is represented by the two bytes `0d 0a` in hexadecimal notation.

#### Tabbed tuples

Tabbed tuples are used in *text/plain* representation when several values are returned, such as tuples or lists. Tabbed tuples delimits each value by the tab character, 09 in hexadecimal notation. Together with *CR LF* it is used to create lists of tuples. Users should ignore any output after the last defined element in the tuple, more elements may be returned in future versions of the API.

## 1.4 Time representation

This section describes how time is handled in the system. There are five main categories related to time which will be discussed here: time bases, time positions (a.k.a. time codes), time intervals, time durations and time span.

### 1.4.1 Time bases

A time base describes how long one unit of time is in seconds using a ratio. This means that everything that has to do with time is done using rational numbers. For instance, ten seconds in the time base used by PAL (1/25) would be 250 units, or 250/25.

#### Textual representations

When working with time bases it is sometimes necessary to construct textual representations which are human readable and can be more easily output and entered into the system. To that end the following textual representations are valid for time bases:

1. Its inverse as a rational number. The syntax is { **denominator** }[:{ **numerator** }], where numerator can be omitted if it's value is one.
2. A *TimeBaseConstant* string

#### TimeBaseConstant

The following time base constants are currently defined:

| | |
|---|---|
| PAL | 1/25 |
| NTSC | 1001/30000 |
| NTSC30 | 1/30 |

#### Examples

1. 25, 30000:1001, 48000
2. PAL, NTSC

#### XSL

*TimeBaseType* is the XML representation of a time base.

```
<xs:complexType name="TimeBaseType">
    <xs:sequence>
        <xs:element name="numerator" type="xs:int"/>
        <xs:element name="denominator" type="xs:int"/>
    </xs:sequence>
</xs:complexType>
```

#### Examples

```
<timeBase>
    <numerator>1</numerator>
    <denominator>25</denominator>
</timeBase>
```

### 1.4.2 Time codes

A time code is a representation of a point in time in some time base.

#### Textual representations

When working with time codes it is sometimes necessary to construct textual representations with are human readable and can be more easily output and entered into the system. To that end the following textual representations are valid for time codes:

1. A sample count and a time base. The syntax is { **number of samples** }[@{ **textual representation of time base** }], where the time base is optional and implicitly one second if omitted. Examples: `124`, `124222@44100`, `400@30000:1001`, `400@NTSC`.
2. A decimal number. Example: `124.25` (will be treated as `12425/100` or `497/4`). This is strongly not recommended, as most sampling frequencies do not have a finite decimal representation!
3. A decimal number and a time base. Example: `124.25/PAL` (will be treated as `12425/2500`). This is also not recommended!

4. The special constants -INF and +INF, representing the earlier than the earliest possible instant and later than the latest possible instant, respectively.

#### XSL

*TimeCodeType* is the XML representation of a time code.

```
<xs:complexType name="TimeCodeType">
    <xs:sequence>
        <xs:element name="samples" type="xs:long"/>
        <xs:element name="timeBase" type="tns:TimeBaseType"/>
    </xs:sequence>
</xs:complexType>
```

#### Examples

```
<timeCode>
    <samples>250</samples>
    <timeBase>
        <numerator>1</numerator>
        <denominator>25</denominator>
    </timeBase>
</timeCode>
```

### 1.4.3 Time intervals

A time interval consists of two time codes: start and end. The time between them denotes the period of time which is of interest. Note that start and end specify an interval like [start,end) in mathematical notation. In other words, the end time code is not within the interval.

Specifying an interval where both time codes have different time bases is valid.

#### XSL

```
<xs:complexType name="TimeIntervalType">
    <xs:sequence>
        <xs:element name="start" type="tns:TimeCodeType"/>
        <xs:element name="end"   type="tns:TimeCodeType"/>
    </xs:sequence>
</xs:complexType>
```

#### Examples

#### Interval in PAL

```
<!-- Seconds 10-20 in PAL -->
<interval>
    <start>
        <samples>250</samples>
        <timeBase>
            <numerator>1</numerator>
            <denominator>25</denominator>
        </timeBase>
    </start>
    <end>
        <samples>500</samples>
```

```
        <timeBase>
            <numerator>1</numerator>
            <denominator>25</denominator>
        </timeBase>
    </end>
</interval>
```

#### Mixed time bases

```
<!-- Approximately seconds 10-20. Start in PALs time base, end in NTSCs time base␣
↪(for instance cutting from PAL to NTSC video) -->
<interval>
    <start>
        <samples>250</samples>
        <timeBase>
            <numerator>1</numerator>
            <denominator>25</denominator>
        </timeBase>
    </start>
    <end>
        <samples>599</samples>
        <timeBase>
            <numerator>1001</numerator>
            <denominator>30000</denominator>
        </timeBase>
    </end>
</interval>
```

### 1.4.4 Time durations

A time duration is the length of a time interval. It can be calculated by subtracting the end time code from the start time code. This means it's simply another time code, with its time line's zero at the start of the interval.

### 1.4.5 Time span

A time span is a interval between two *time codes*.

There are two notations. The first notation is by using two time instants and separate them with a hyphen (-). The first time instant is included in the interval, the second one is excluded. That is, in the interval 124-221, the instant corresponding to second 124 is included in the interval, but not the instant corresponding to second 221. (E.g., if there is an instant corresponding to second 220.9999999, it is included.)

The other notation is by using one time instant and one *time duration*, separate with a plus sign (+). The notation { **a** } + { **b** } is equivalent to { **a** } − { **a** + **b** }.

## 1.5 Content paths

Content paths are accepted by certain resources as a way of controlling the data that is returned for one or more entity.

### 1.5.1 Paths

A path consists of a list of keys that correspond to an field in the response. For example:

```
shape.containerComponent.format
shape.containerComponent.duration
```

```
shape.containerComponent.file.path
shape.containerComponent.file.size
```

Paths can also be written using a short-hand format:

```
shape.containerComponent.[format,duration]
shape.containerComponent.file.[path,size]
```

A path that selects from an element that represents a sequence will select on all elements in the sequence. This path selects the codec from all audio components for example:

```
shape.audioComponent.codec
```

Keys can contains a qualifier that further restricts the response:

```
shape[tag=original].containerComponent.[format,duration]
shape[tag=original].containerComponent.file.[path,size]
```

The syntax is:

```
path        ::=   key ( ''.''  key )*
key         ::=   identifier (``['' qualifier ``]'')*
qualifier   ::=   identifier ``='' identifier
identifier  ::=   letter { letter | number }*
```

#### Example

Fetch all metadata fields and groups:

```
p=metadata.timespan.[field,group]
```

Fetch all metadata groups:

```
p=metadata.timespan.group
```

Fetch only "second level" groups:

```
p=metadata.timespan.group.group
```

Fetch the contents of a group called "test_group":

```
p=metadata.timespan.group[name=test_group]
```

Fetch child metadata fields in a group called "test_group":

```
p=metadata.timespan.group[name=test_group].field
```

Fetch only the `name` and `value` of metadata fields (excluding properties like `uuid`, `timestamp`, etc.):

```
p=metadata.timespan.field.[name,value].value
```

### 1.5.2 Aliases

Aliases can be used to shorten long path strings and to refer to multiple paths at once. Aliases can have arguments, making them similar to macros in other programming languages.

```
alias   ::=   name [ ``('' arg (','' arg)* '')'' ] ``='' path (','' path)*
name    ::=   letter { letter | number }*
arg     ::=   letter { letter | number }*
```

As with fields and configuration properties, prefer to prefix alias names with a unique application prefix, to avoid possible conflicts in the future.

When an alias is evaluated, any arguments in the path, expressed as `$arg`, will be replaced by the argument value. The resulting path string must be a valid path. For example, an alias that provides information about a shape:

```
detail(tag)=shape[tag=$tag].containerComponent.format,shape[tag=$tag].videoComponent.
↪[codec,duration]
```

Configure aliases using the *path alias configuration resource*.

```
PUT /configuration/path-alias
Content-Type: application/xml

<PathAliasConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <alias>v(name)=metadata.timespan[start=-INF][end=+INF].field[name=$name].value.value
↪</alias>
  <alias>detail(tag)=shape[tag=$tag].containerComponent.format,shape[tag=$tag].
↪videoComponent.[codec,duration]</alias>
</PathAliasConfigurationDocument>
```

### Default aliases

**v(name)** `metadata.timespan[start=-INF][end=+INF].field[name=$name].value.value`

---

**Note:** Transient metadata will not be returned using content paths, unless explicitly specified in the path. For example, `metadata` or `metadata.timespan` give no transident field. And pathes like `v(transient_field)` should be used for transient fields to be returned.

---

## 1.6 Constants

An assorted list of constants found in the Vidispine API.

- *Job states*
- *Job types*
- *Storage states*
- *Storage types*
- *File States*
- *System configuration*

CHAPTER

TWO

# ITEMS AND METADATA

This chapter describes the Item, the central entity in the Vidispine data model, and how metadata (information about the item) can be associated with the item.

## 2.1 Imports

Importing is the process of registering essence/media with Vidispine. As Vidispine works with files and objects, either local or in the cloud, another way of putting it is to say that the media file(s) become under Vidispine's supervision.

**Note:** Vidispine does not support machine control or baseband video ingest (http://en.wikipedia.org/wiki/Serial_digital_interface) . However, growing files are supported, including operations on items that are currently being ingested.

### 2.1.1 Importing items

There are several ways of importing media into Vidispine. Which one that is used depends on where the media is located, the order of operations, and what automation that is required.

On a high level, the different ways of importing are:

**Regular import** This import uses a URI pointing outside of Vidispine storages to reference the source media. Vidispine will make a copy of the source material (and sidecar files given by the user) to a Vidispine storage.

The job type for this type of import is PLACEHOLDER_IMPORT. For reference information, see *Import using a URI*.

**Raw import** Here, the caller supplies the material in the REST API call as data in the request body. This is useful when the data is stored as a file at the caller's point, for example when the end user is uploading information in a web browser. It is also useful when the information resides in a location which Vidispine cannot reach, for example behind a firewall.

Vidispine supports partial upload, so the caller can split the input in multiple parts in order to better handle network problems or in order to parallelize uploads.

The job type for this type of import is RAW_IMPORT. Note that the job is not created until all parts of the file has been uploaded. For reference information, see *Import using the request body*.

**File import** This import is used where the file is already located on a storage which is supervised by Vidispine. In this type of import, no copying takes place. Instead, a new item is created, and the file is associated with the file.

The job type for this type of import is RAW_IMPORT. For reference information, see *Importing a file from a storage*.

**Auto-import** This is a special case of file import, where no explicit call has to be done for every file. The user sets up *rules* for how files are imported, and if any sidecar files are processed as well.

The job type for this type of import is AUTO_IMPORT. For reference information, see *Auto-import rules*.

**Placeholder import** A placeholder import is an import where the placeholder item and a placeholder shape are created before any file is imported. When creating the placeholder shape, the caller gives item metadata and information about the components. The creation of the placeholder item is a synchronous operation, and the item id is immediately returned.

Using the item id, the caller can populate the placeholder shape with files, either by posting the URI or the raw content to the components of the shape. The placeholder import is the import method that gives the highest flexibility.

For reference information, see *Placeholder imports*.

#### Sidecar files

Sidecar files, containing metadata, subtitles, or other supplementary information, can be imported to an item either at the same time as the item is imported, or afterwards using an sidecar import job.

### 2.1.2 Steps of import operation

Every import job consists of a number of job steps. Some of the the job steps run in parallel, and some in sequence. These are the most important steps in an import job:

- Create entities. The item and the *original* shape is created. This will not take place if the caller already has created the item before the job (placeholder import).
- Transfer media. The media is transferred from the source URI to a Vidispine storage. This will not take place if the media is already located on a Vidispine storage (raw import, file import, auto-import).
- Initial media check. The media is checked using a *shape deduction* by the transcoder. The components of the original shape are created.
- Transcoding. Using the information about the original shape and all shape tags given by the caller at the invocation of the job, a transcoding task is created and given to the transcoder.
- A media check of all new shapes takes place, as soon as the transcoder has started to work.
- A final media check of the original media and transcoder shapes is done after the transcoder has finished.
- Any XMP, EXIF, or document metadata is extracted.
- Optionally, the original shape can be replaced by a transcoded shape. This is useful if one seeks to have one "house format" as the original shape format, and all incoming material of other types should be converted into the house format.
- Sidecar files are imported.

### 2.1.3 Transcoding

During import, the caller decides which shape tags that are to be created from the original media. By default, thumbnails are created according the to the shape tag definitions. The caller can choose which thumbnail service resource to use, if multiple resources are set up. This is done using the query parameter `thumbnailService`. In addition to thumbnails, full-resolution posters images can be created, by supplying a list of timecodes in the query parameter `createPosters`. The creation of thumbnails can be disabled by setting the query parameter `thumbnails` to false.

### 2.1.4 Notifications

As with all jobs in Vidispine, the caller can be notified about the job progress by HTTP messages or other actions. This is described in the *Notifications* section.

### 2.1.5 Adjusting import

The import API is very rich and contains several parameters. Fortunately, most of the time, the default values can be used.

#### Import settings

Settings that are used during imports can be set prior to starting an import job. An example of such a setting are access control lists. The settings can then be used by specifying the id of the settings profile using the query parameter `settings`.

#### Special job metadata values

Special instructions can be supplied to the import job via the the query parameter `jobmetadata=` { **key** = **value** }. Note that the equals sign is part of the value of the query parameter, so it has to be URL encoded (`%3d`)

#### Cut off start and end of video

Given that the video has SMPTE timecodes, an interval can be cut out using the metadata *`smpteTimeCode`* and *`lastSmpteTimeCode`*. If the video has no SMPTE timecode, the interval is calculated based on the first timestamp in the video.

#### Checksum on file transfer

Normally, the checksum of the imported files will be computed asynchronously in the background. For `PLACEHOLDER_IMPORT` jobs, by specifying the `jobmetadata checksumMode%3Dtransfer`, the checksum of files will be computed during the transfer step of the job. See *`checksumMode`*.

## 2.2 Exports

An item export is the process of copying a file from storage to a location accessible by the system.

### 2.2.1 Exporting items

Exporting the files of an item is an asynchronous operation that is performed by an `EXPORT` job. The *export resource* allows you to:

- Export files for an item.
- Export files for the items in a specific collection or library.
- Export files for specific shapes only.
- Export partial file content, by specifying a start and end time code.

There are a number of operations that can be performed as part of an export. An export job can:

- Restore files from archive if necessary.
- Transcode into the selected formats.
- Rewrite the XMP in the exported files so that it matches the XMP in the item metadata.
- Create a sidecar XML file containing the item metadata.
- Transfer the files to the final location.

#### Example

To export the original shape of a specific item to a directory on the local file system:

```
POST /item/VX-191440/export?uri=file:///srv/exported/&tag=original
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-169822</jobId>
  <user>admin</user>
  <started>2014-07-03T09:39:52.969Z</started>
  <status>READY</status>
  <type>EXPORT</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

### 2.2.2 Export locations

It is possible to pre-define named export locations. When starting an export job, the location name can be passed as a parameter, the files will then be exported to the URI associated with the export location.

```
PUT /API/export-location/default-exports
Content-Type: application/xml

<ExportLocationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>file:///srv/exported/</uri>
</ExportLocationDocument>
```

```
POST /item/VX-191440/export?locationName=default-exports&tag=original
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-169824</jobId>
  <user>admin</user>
  <started>2014-07-03T09:49:12.972Z</started>
  <status>READY</status>
  <type>EXPORT</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

See the *export location resource* for more information.

#### File naming scripts

Export locations can have a JavaScript associated with them. These work the same way as file name scripts on storages (see *Naming files on storage*). The difference is that for export locations, the script will not be retried if there is a filename conflict. That is, if the filename generated by the script is already taken, then the existing file will be overwritten.

There are two ways to add a script to an export location, either using XML, or by using the *script resource*.

#### Example

Adding a script to an export location using XML.

```
PUT /export-location/External_FTP HTTP/1.1
Content-Type: application/xml

<ExportLocationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
```

```
  <uri>ftp://user:password@10.2.23.25/export/</uri>
  <script>filename = context.getOriginalFilename() + "." + context.getExtension();</
↪script>
</ExportLocationDocument>
```

#### Example

Adding a script to an export location using the `script` REST resource.

```
PUT /export-location/External_FTP/script HTTP/1.1
Content-Type: text/plain

filename = context.getOriginalFilename() + "." + context.getExtension();
```

### 2.2.3 Export templates

**Developer preview**

Export templates are a way of expressing complex export content. That is, the exact structure of the files exported can be described by an XML or JSON structure. See export-templates.

## 2.3 Item metadata

The metadata of an item consists of fields, groups and values that belong to a specific interval or timespan. Metadata that does not apply to a specific interval, that is, it is non-timed, belong to the timespan with a start and end of `-INF` and `+INF`, respectively.

- A timespan describes an interval within the item, denoted by two *time codes* (a start value and an end value).
- A timespan contains sets of fields and groups.
- Groups are named sets of fields and groups.
- Fields have a *name* and a set of values of a specific type.

In addition, metadata can apply to a specific component track, identified by a track name, for example, A1 or V1 (a name on the regular expression form `[A-Za-z]+[1-9][0-9]*`), and a specific language, typically a ISO 639 language code.

Examples of usage can be found at *Creating fields/groups, modifying and moving metadata*.

### 2.3.1 Fields

Before you can use fields and groups in the metadata of an item you need to define them. When defining a field you must select its data type, that is, the type of values that will be accepted for the field. You can also restrict values further by adding additional *restrictions* to the field.

```
PUT /metadata-field/event_type HTTP/1.1
Content-Type: application/xml

<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <type>string-exact</type>
    <stringRestriction>
      <pattern>[a-z]+</pattern>
    </stringRestriction>
</MetadataFieldDocument>
```

```
HTTP/1.1 200 OK
```

### Field identifiers

Metadata field ids are case sensitive and must have a certain format to avoid conflicts with existing and possible future fields used by Vidispine or other partners.

A metadata field id (name) is one of:

- **Core set**, the standard metadata set. Metadata field ids are assigned by Vidispine, and are of the regular expression form: `[A-Za-z][A-Za-z0-9]*`. maximum 32 characters.
- **Common set**. Metadata field ids have the form { *category* } _ { *field-name* }. The *category* is of the regular expression form: `[A-Za-z][A-Za-z0-9]*`, maximum 4 characters, and assigned by Vidispine to be used by industry partners. *field-name* is the regular expression form: `[A-Za-z][A-Za-z0-9]*`. Total length of id is maximum 32 characters, including the underscore ( _ ) character.
- **Custom set**. Metadata field ids have the form { *custom-name* } _ { *field-name* }. The *custom-name* is of the regular expression form: `[A-Za-z][A-Za-z0-9]*`, *minimum* 5 characters, and assigned by Vidispine. *field-name* is the regular expression form: `[A-Za-z][A-Za-z0-9]*`. Total length of id is maximum 32 characters, including the underscore ( _ ) character.

### Data types

The data types at your disposal are:

| Data type | Description |
|---|---|
| `date` | An ISO-8601 compatible timestamp. |
| `float` | A floating point value. |
| `integer` | A 32-bit signed integer value. |
| `long` | A 64-bit signed integer value. |
| `string` | A string. |
| `string-exact` | A string that uses exact matching. |
| `boolean` | A boolean value. |
| `timeCode` | A *time code* value. |

### string vs string-exact

During index time, the value of a `string` field is broken into small tokens, and then processed by various filters before been indexed. By doing so, users would get nice phrase search results, but loose the ability of "exact match".

The value of a `string-exact` field, on the other hand, is indexed directly as a single token. This makes a "exact match" possible, and leads to smaller index size.

Note: In order to make search working properly, a re-index is required if the filed type is changed.

### Noindex-types

Deprecated since version 4.1: The `index` element on the metadata field should be used instead to control if a field should be indexed.

Use the `noindex` types for fields that will contain data that should not be indexed, for example if it will never be searched for or if it contains data in some format, for example JSON or Base64-encoded binary data.

| Data type | Description |
|---|---|
| `date-noindex` | An ISO-8601 compatible timestamp. No indexing will take place. |
| `float-noindex` | A floating point value. No indexing will take place. |
| `integer-noindex` | A 32-bit signed integer value. No indexing will take place. |
| `long-noindex` | A 64-bit signed integer value. No indexing will take place. |
| `string-noindex` | A string. No indexing will take place. |
| `boolean-noindex` | A boolean value. No indexing will take place. |
| `timeCode-noindex` | A *time code* value. No indexing will take place. |

### Sortable types

Deprecated since version 3.2: Sortable types are deprecated. This is since any field type can be used for sorting as long as it is indexed.

Sortable types can be used when searching to sort search results. A sortable field is one that uses a sortable types. Fields that are sortable have two limitations:

1. They can only exist within non-timed metadata.
2. They cannot contain lists of values.

| Data type | Description |
|---|---|
| `date-sortable` | An ISO-8601 compatible timestamp. Can be used for sorting. |
| `float-sortable` | A floating point value. Can be used for sorting. |
| `integer-sortable` | An integer value. Can be used for sorting. |
| `string-sortable` | A string. Can be used for sorting. |
| `string-exact-sortable` | A string that uses exact matching. Can be used for sorting. |

### Restrictions

Add restrictions to metadata fields for further restrict the values that are to be allowed for a field. The table below shows the different types of restrictions that exist.

| Data type | Parameter | Restriction |
|---|---|---|
| `string` | `pattern` | A Java compatible regular expression (http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html) |
| | `minLength` | A minimum allowed length of the string. |
| | `maxLength` | A maximum allowed length of the string. |
| `float` | `minInclusive` | A minimum allowed value (inclusive). |
| | `maxInclusive` | A maximum allowed value (inclusive). |
| `integer` | `minInclusive` | A minimum allowed value (inclusive). |
| | `maxInclusive` | A maximum allowed value (inclusive). |
| `long` | `minInclusive` | A minimum allowed value (inclusive). |
| | `maxInclusive` | A maximum allowed value (inclusive). |

For example, adding a field that only accept integer values in the interval [1, 5].

```
PUT /metadata-field/event_rating HTTP/1.1
Content-Type: application/xml

<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>integer</type>
  <integerRestriction>
    <minInclusive>1</minInclusive>
    <maxInclusive>5</maxInclusive>
  </integerRestriction>
</MetadataFieldDocument>
```

**Note:** The naming of your field must follow certain rules, see *Field identifiers*.

### Default values

You can assign a default value to a field if you want a field to be included when retrieving the metadata of an item even if it has not been set.

Default values are added to the search index, meaning that searching for fields by the default value is possible.

**Note:** If the default value is changed then the search index should be rebuilt using :http:put::*/reindex/(index)*, or, the relevant items manually re-indexed using :http:put::*/item/(item-id)/re-index*.

```
PUT /metadata-field/testing_default HTTP/1.1
Content-Type: application/xml

<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>integer</type>
  <defaultValue>0</defaultValue>
</MetadataFieldDocument>
```

```
HTTP/1.1 200 OK
```

Use the `defaultValue` parameter to control if the field should be included with the default value. Here item VX-12 does not have the field set:

```
GET /item/VX-12/metadata?field=testing_default&defaultValue=false HTTP/1.1
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <item id="VX-12">
    <metadata>
      <revision>VX-59,VX-60,VX-57</revision>
    </metadata>
  </item>
</MetadataListDocument>
```

```
GET /item/VX-12/metadata?field=testing_default&defaultValue=true HTTP/1.1
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <item id="VX-12">
    <metadata>
      <revision>VX-59,VX-60,VX-57</revision>
      <timespan end="+INF" start="-INF">
        <field>
          <name>testing_default</name>
          <value>0</value>
        </field>
      </timespan>
    </metadata>
  </item>
</MetadataListDocument>
```

### 2.3.2 Field groups

Metadata fields can be organized in zero or more *field groups*. Use groups to represents events or other types of objects in the metadata.

```
PUT /metadata-field/field-group/event HTTP/1.1
Content-Type: application/xml

<MetadataFieldGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <data>
    <key>description</key>
    <value>An event in a clip</value>
  </data>
  <field>
    <name>event_type</name>
    <data>
      <key>text</key>
      <value>Here is some text.</value>
    </data>
  </field>
  <field>
    <name>event_rating</name>
  </field>
  <field>
    <name>event_text</name>
    <type>string</type>
    <data>
      <key>someextradata</key>
      <value>Some additional data</value>
    </data>
  </field>
  <access>
    <user>admin</user>
    <permission>DELETE</permission>
  </access>
</MetadataFieldGroupDocument>
```

Fields in a group that have not yet been created will be created for you. The example above also shows how additional metadata can be added to fields and groups.

### 2.3.3 Metadata schema

Finally, you can define a *metadata schema* to make sure that the metadata conforms to a specific data model.

For an example of how to define a metadata schema, see *Defining a metadata schema*. You can also define the schema when creating field groups, as shown in *Alternate way of creating a schema*.

There are three different types of elements in the schema: groups, fields and nested groups. They all have in common three attributes, `name`, `min` and `max`, and the two latter elements also have the attribute `reference`.

- **Name** is the name of the field or group that the element refers to. The table below shows the semantics of a property for the different elements.
- **Min** specifies the minimum of times that the element can occur in that context and is a non-negative integer.
- **Max** specifies the maximum of times that the element can occur in that context and if set to a negative value it will be interpreted as an infinite number of times.

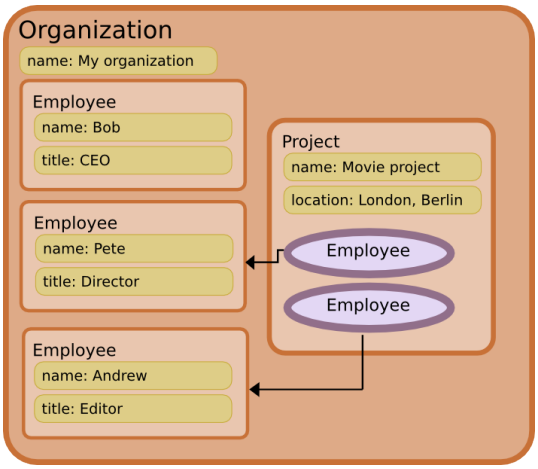
| | Group | Nested groups | Field |
|---|---|---|---|
| min | The minimum number of times that the group can occur at top-level. | The minimum number of times that the group can occur inside the given group | The minimum number of times the field can occur inside the given group |
| max | The maximum number of times that the group can occur at top-level. | The maximum number of times that the group can occur inside the given group | The maximum number of times the field can occur inside the given group |
| name | The name of the group. | The name of the group. | The name of the field. |
| reference | - | If set, controls whether the group must be a reference or not. | If set, controls whether the group must be a reference or not. |

Top-level groups are used to specify what a fields and groups that they are allowed to contain. Furthermore they specify whether or not that group can exist outside of other groups. Nested groups and fields are used to specify the content of a top-level group.

### 2.3.4 Hierarchical metadata

Complex data relations can be represented with hierarchical metadata. Let's say we have three classes in our data model, Organization, Employee and Project. An organization has a name, one or more employees and one or more projects. An employee has a name and a title. A project has a name and one or more employees assigned to it. This data model can be represented by using *field groups* to represent the classes and *fields* to represent the attributes.

Below an example of this data model is given:




As can be seen in the diagram, weak references are used in the project to point to the employees in the organization to avoid data duplication. An equivalent XML of the above diagram:

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    <group>
      <name>organization</name>
      <field>
        <name>name</name>
        <value>My organization</value>
      </field>
      <group uuid="c9be268e-03f4-4378-8061-e1c8b8f6b45c">
        <name>employee</name>
        <field>
          <name>name</name>
          <value>Bob</value>
        </field>
```

```
        <field>
          <name>title</name>
          <value>CEO</value>
        </field>
      </group>
      <group uuid="96a333b1-06f0-4975-adee-78b93c2a7614">
        <name>employee</name>
        <field>
          <name>name</name>
          <value>Pete</value>
        </field>
        <field>
          <name>title</name>
          <value>Director</value>
        </field>
      </group>
      <group uuid="82f92192-d2ef-422a-984a-b03cb0476a8a">
        <name>employee</name>
        <field>
          <name>name</name>
          <value>Andrew</value>
        </field>
        <field>
          <name>title</name>
          <value>Editor</value>
        </field>
      </group>
      <group>
        <name>project</name>
        <field>
          <name>name</name>
          <value>Movie project</value>
        </field>
        <field>
          <name>location</name>
          <value>London</value>
          <value>Berlin</value>
        </field>
        <group>
          <name>employee</name>
          <reference>96a333b1-06f0-4975-adee-78b93c2a7614</reference>
        </group>
        <group>
          <name>employee</name>
          <reference>82f92192-d2ef-422a-984a-b03cb0476a8a</reference>
        </group>
      </group>
    </group>
  </timespan>
</MetadataDocument>
```

### 2.3.5 Metadata inheritance

Metadata fields and groups can be marked as *inheritable*. Inheritable metadata set on a collection will be inherited to all items in the collection. If an inheritable field is set on the collection *and* the item, the value on the item takes precedence.

To enable metadata inheritance add the attribute `inheritance="true"` to the root element of the MetadataField-

Document:

```
PUT /metadata-field/event_rating HTTP/1.1
Content-Type: application/xml

<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine" inheritance=
↪"true">
  <type>integer</type>
  <integerRestriction>
    <minInclusive>1</minInclusive>
    <maxInclusive>5</maxInclusive>
  </integerRestriction>
</MetadataFieldDocument>
```

When retrieving an item with inherited metadata the attribute `inheritance` on the `field` element will contain the ID of the collection from which the field was inherited:

```
GET /item/VX-51/metadata HTTP/1.1
Accept: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <item id="VX-51">
        <metadata>
            <revision>VX-206,VX-146,VX-145,VX-154</revision>
            <timespan start="-INF" end="+INF">
                ...
                <field inheritance="Collection/VX-6" uuid="9beea1d8-0560-40c2-bfaf-
↪03ff2dd5f02e" user="admin" timestamp="2018-03-14T13:23:14.238+01:00" change="VX-148
↪">
                    <name>event_rating</name>
                    <value uuid="593e6678-1ace-4d69-9362-1add839024bf" user="admin"␣
↪timestamp="2018-03-14T13:23:14.238+01:00" change="VX-148">3</value>
                </field>
                ...
            </timespan>
        </metadata>
    </item>
</MetadataListDocument>
```

### Relative timespan inheritance

New in version 5.0.

By setting the `absoluteTime` attribute in the CollectionDocument, timespans inherited from a collection are individually offset by the *startTimeCode* value of child items. In other words, timespans set on the collection are interpreted as absolute, and converted to the usual relative (zero-based) timecodes for each item.

### Example

To create a collection using this feature:

```
POST /collection?name=Sports-Game
Accept: application/xml

<CollectionDocument absoluteTime="true" xmlns="http://xml.vidispine.com/schema/
↪vidispine">
</CollectionDocument>
```

If this collection contains an item with metadata field `startTimeCode` with value 100@PAL:

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
        <timespan start="-INF" end="+INF">
            <field>
                <name>startTimeCode</name>
                <value>100@PAL</value>
            </field>
        </timespan>
</MetadataDocument>
```

and the collection metadata contains an inheritable field, in this case a *custom field* called `goal`:

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
        <timespan start="500@PAL" end="1000@PAL">
            <field>
                <name>goal</name>
                <value>by John Smith</value>
            </field>
        </timespan>
</MetadataDocument>
```

The metadata on the item then shows the time of the inherited field relative to its `startTimeCode` value:

```
GET /item/VX-1/metadata
```

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    ...
    <timespan end="400@PAL" start="900@PAL">
        <field>
            <name>goal</name>
            <value>by John Smith</value>
        </field>
    </timespan>
    ...
</MetadataDocument>
```

The default behaviour when `absoluteTime` is `false` or not set, or no `startTimeCode` is set on the item, is that that timespans are inherited unchanged from the collection.

### 2.3.6 Versioning

Metadata essentially consists of key-value pairs. The key of a value is its UUID, but can also often be described by the quintuple (timespan, group, field name, track, language). However the latter does not guarantee unambiguity. If at any point a key corresponds to more than one value, then a conflict exists.

#### Change sets

A change set is a set of changes to the metadata. The change set has a unique id and can be related to other change sets. The current revision of the metadata is essentially the superset of all change sets.

#### Example

If we start with a newly imported item, its metadata might look like this:

```
GET item/VX-250/metadata
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <item id="VX-250">
    <metadata>
      <revision>VX-30</revision>
      <timespan end="+INF" start="-INF">
        <field>
          <name>durationSeconds</name>
          <value change="VX-30" timestamp="2010-03-19T09:08:09.563+01:00" user="system
↪">232.32</value>
        </field>
        <field>
          <name>durationTimeCode</name>
          <value change="VX-30" timestamp="2010-03-19T09:08:09.576+01:00" user="system
↪">232320000@1000000</value>
        </field>
      </timespan>
    </metadata>
  </item>
</MetadataListDocument>
```

Assume two users, u1 and u2, both wants to add a title, not knowing of each others changes.

```
PUT item/VX-250/metadata?revision=VX-30
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan end="+INF" start="-INF">
    <field>
      <name>title</name>
      <value>u1's title</value>
    </field>
  </timespan>
</MetadataDocument>
```

```
PUT item/VX-250/metadata?revision=VX-30
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan end="+INF" start="-INF">
    <field>
      <name>title</name>
      <value>u2's title</value>
    </field>
  </timespan>
</MetadataDocument>
```

The result of the two operations will result in a conflict, because u2 did not know of the change made by u1.

```
GET item/VX-250/metadata
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item>
      <metadata>
         <revision>VX-30,VX-32,VX-31</revision>
         <timespan end="+INF" start="-INF">
            <field conflict="true">
               <name>title</name>
```

```
                <value change="VX-32" timestamp="2010-03-19T09:16:56.419+01:00" user=
↪"u2">u2's title</value>
                <value change="VX-31" timestamp="2010-03-19T09:16:25.454+01:00" user=
↪"u1">u1's title</value>
            </field>
            <field>
                <name>durationSeconds</name>
                <value change="VX-30" timestamp="2010-03-19T09:08:09.563+01:00" user=
↪"system">232.32</value>
            </field>
            <field>
                <name>durationTimeCode</name>
                <value change="VX-30" timestamp="2010-03-19T09:08:09.576+01:00" user=
↪"system">232320000@1000000</value>
            </field>
        </timespan>
    </metadata>
  </item>
</MetadataListDocument>
```

In order to resolve the conflict u1 inserts another change set:

```
PUT item/VX-250/metadata?revision=VX-30,VX-32,VX-31
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan end="+INF" start="-INF">
      <field>
         <name>title</name>
         <value>u1's and u2's title</value>
      </field>
   </timespan>
</MetadataDocument>
```

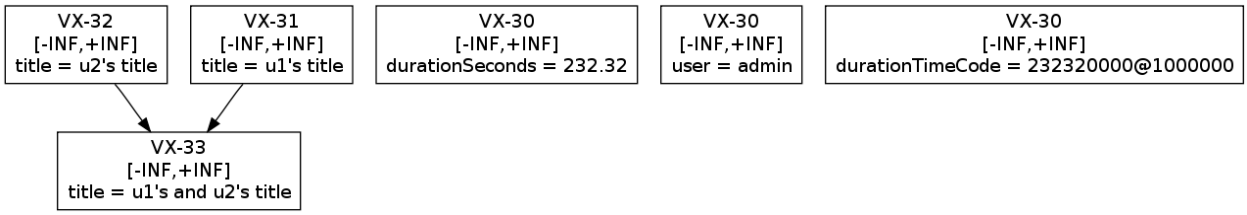
Which results in:

```
GET item/VX-250/metadata
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item>
      <metadata>
         <revision>VX-30,VX-33</revision>
         <timespan end="+INF" start="-INF">
            <field>
                <name>title</name>
                <value change="VX-33" timestamp="2010-03-19T09:21:28.692+01:00" user=
↪"u1">u1's and u2's title</value>
            </field>
            <field>
                <name>durationSeconds</name>
                <value change="VX-30" timestamp="2010-03-19T09:08:09.563+01:00" user=
↪"system">232.32</value>
            </field>
            <field>
                <name>durationTimeCode</name>
                <value change="VX-30" timestamp="2010-03-19T09:08:09.576+01:00" user=
↪"system">232320000@1000000</value>
            </field>
```

```
            </timespan>
        </metadata>
    </item>
</MetadataListDocument>
```

A graph of this can be seen below. Worth noting is that it is the leaves of the graph that represent the current revision.




## 2.3.7 Structure of metadata

### Lists of values

A field can contain multiple values.

### Example

Retrieving the current metadata:

```
GET /item/VX-250/metadata
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <item id="VX-7612">
        <metadata>
            <revision>VX-16113,VX-16114</revision>
            <timespan end="+INF" start="-INF">
                <field change="VX-16114" timestamp="2010-08-16T08:28:18.592+02:00" user=
↪"system" uuid="4cc88be0-4fc3-4243-a6e0-b1a151e6cde0">
                    <name>shapeTag</name>
                    <value change="VX-16114" timestamp="2010-08-16T08:28:18.592+02:00"␣
↪user="system" uuid="b98a5553-a6ca-4235-bb14-fc17fdf7eda3">original</value>
                </field>
                <field change="VX-16113" timestamp="2010-08-16T08:28:18.366+02:00" user=
↪"admin" uuid="d35fb0ea-cd05-4429-a707-b248420b3fe7">
                    <name>field_a</name>
                    <value change="VX-16113" timestamp="2010-08-16T08:28:18.366+02:00"␣
↪user="admin" uuid="31602cd8-4cfa-4912-a6fb-d731841f880c">my value</value>
                </field>
            </timespan>
        </metadata>
    </item>
</MetadataListDocument>
```

Adding a new value to `field_a`, if the `mode` attribute is left out the existing value will be modified instead of adding it as a new value.

```
PUT /item/VX-250/metadata
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <timespan start="-INF" end="+INF">
```

```
      <field>
         <name>field_a</name>
         <value mode="add">my other value</value>
      </field>
   </timespan>
</MetadataDocument>
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item>
      <metadata>
         <revision>VX-16113,VX-16114,VX-16115</revision>
         <timespan end="+INF" start="-INF">
            <field change="VX-16115" timestamp="2010-08-16T08:35:18.550+02:00" user=
↪"admin" uuid="d35fb0ea-cd05-4429-a707-b248420b3fe7">
               <name>field_a</name>
               <value change="VX-16113" timestamp="2010-08-16T08:28:18.366+02:00"␣
↪user="admin" uuid="31602cd8-4cfa-4912-a6fb-d731841f880c">my value</value>
               <value change="VX-16115" timestamp="2010-08-16T08:35:18.550+02:00"␣
↪user="admin" uuid="cb47404c-5d69-466e-ad61-733b2cf8496b">my other value</value>
            </field>
            <field change="VX-16114" timestamp="2010-08-16T08:28:18.592+02:00" user=
↪"system" uuid="4cc88be0-4fc3-4243-a6e0-b1a151e6cde0">
               <name>shapeTag</name>
               <value change="VX-16114" timestamp="2010-08-16T08:28:18.592+02:00"␣
↪user="system" uuid="b98a5553-a6ca-4235-bb14-fc17fdf7eda3">original</value>
            </field>
         </timespan>
      </metadata>
   </item>
</MetadataListDocument>
```

In order to modify either of the two values of the field the UUID must be specified, otherwise ambiguity will exist.

```
PUT /item/VX-250/metadata
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan start="-INF" end="+INF">
      <field>
         <name>field_a</name>
         <value>my new value</value>
      </field>
   </timespan>
</MetadataDocument>
```

```
400 An invalid parameter was entered
Context: metadata
Reason: Ambiguous path to value
```

Values can be removed by setting the `mode` attribute to `remove`.

```
PUT /item/VX-250/metadata
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan start="-INF" end="+INF">
      <field>
```

```
        <name>field_a</name>
        <value mode="remove" uuid="31602cd8-4cfa-4912-a6fb-d731841f880c"/>
      </field>
   </timespan>
</MetadataDocument>
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item>
      <metadata>
         <revision>VX-16114,VX-16115,VX-16117</revision>
         <timespan end="+INF" start="-INF">
            <field change="VX-16117" timestamp="2010-08-16T08:48:21.474+02:00" user=
↪"admin" uuid="d35fb0ea-cd05-4429-a707-b248420b3fe7">
               <name>field_a</name>
               <value change="VX-16115" timestamp="2010-08-16T08:35:18.550+02:00"␣
↪user="admin" uuid="cb47404c-5d69-466e-ad61-733b2cf8496b">my other value</value>
            </field>
            <field change="VX-16114" timestamp="2010-08-16T08:28:18.592+02:00" user=
↪"system" uuid="4cc88be0-4fc3-4243-a6e0-b1a151e6cde0">
               <name>shapeTag</name>
               <value change="VX-16114" timestamp="2010-08-16T08:28:18.592+02:00"␣
↪user="system" uuid="b98a5553-a6ca-4235-bb14-fc17fdf7eda3">original</value>
            </field>
         </timespan>
      </metadata>
   </item>
</MetadataListDocument>
```

### Weak references

Groups and fields can refer to other groups and fields by using weak references. Furthermore the metadata of other items and collections as well as global metadata can be referenced.

### Example: referencing global metadata

Adding some global metadata:

```
PUT /metadata
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    <group mode="add">
      <name>test</name>
      <field>
        <name>example_name</name>
        <value>Global name</value>
      </field>
    </group>
  </timespan>
</MetadataDocument>
```

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <revision>VX-76,VX-82,VX-80,VX-84</revision>
  <timespan start="-INF" end="+INF">
    <group uuid="aaf7fde8-308d-4555-8a8b-8954f5ec5fd9" user="admin" timestamp="2010-
↪12-27T09:40:32.667+01:00" change="VX-84">
```

```
      <name>test</name>
      <field uuid="376e831b-8e8e-4c0a-a7b2-dfdbb49d2e20" user="admin" timestamp="2010-
↪12-27T09:40:32.667+01:00" change="VX-84">
        <name>example_name</name>
        <value uuid="431d8078-fb05-42f0-87ae-a9ea73b8c4d1" user="admin" timestamp=
↪"2010-12-27T09:40:32.667+01:00" change="VX-84">Global name</value>
      </field>
    </group>
  </timespan>
</MetadataDocument>
```

Referencing it from an item:

```
PUT /item/VX-15/metadata
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    <group mode="add">
      <name>test</name>
      <reference>aaf7fde8-308d-4555-8a8b-8954f5ec5fd9</reference>
    </group>
  </timespan>
</MetadataDocument>
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <item id="VX-15">
    <metadata>
      <revision>VX-86,VX-87</revision>
      <timespan end="+INF" start="-INF">
        <field change="VX-86" timestamp="2010-12-27T09:44:43.594+01:00" user="system"␣
↪uuid="154c5b1c-575d-42f8-947d-5b0d38a78e96">
          <name>shapeTag</name>
          <value change="VX-86" timestamp="2010-12-27T09:44:43.594+01:00" user="system
↪" uuid="eb05a782-75f1-4e42-9e5f-4d93be6f4247">original</value>
        </field>
        <group change="VX-87" timestamp="2010-12-27T09:45:21.786+01:00" user="admin"␣
↪uuid="7c3d0b12-9b0a-48b8-b603-67fdcc26108d">
          <name>test</name>
          <referenced id="" type="global" uuid="aaf7fde8-308d-4555-8a8b-8954f5ec5fd9"/
↪>
          <field change="VX-84" timestamp="2010-12-27T09:40:32.667+01:00" user="admin
↪" uuid="376e831b-8e8e-4c0a-a7b2-dfdbb49d2e20">
            <name>example_name</name>
            <value change="VX-84" timestamp="2010-12-27T09:40:32.667+01:00" user=
↪"admin" uuid="431d8078-fb05-42f0-87ae-a9ea73b8c4d1">Global name</value>
          </field>
        </group>
      </timespan>
    </metadata>
  </item>
</MetadataListDocument>
```

### 2.3.8 Metadata defined by the systems

| Name | Description |
|---|---|
| `user` | The name of the user that imported the item. (Removed in 4.14, see below.) |
| `title` | The title of the item. |
| `shapeTag` | A shape tag that is used on a shape belonging to the item. |
| `representativeThumbnail` | A thumbnail that is representative of the item. Initially set by the system, but can be modified by a user. |
| `representativeThumbnailNoAuth` | Same as above, with the exception that no authentication is required. |
| `itemId` | The id of the item. |
| `mediaType` | The type of the media, e.g. video/audio/image/document/pdf/binary. |
| `mimeType` | The mime type of the media, e.g. image/jpeg. |
| `created` | The time when item was created. |
| `startTimeCode` | The start time of the item expressed as a time code. |
| `startSeconds` | The start time of the item expressed in seconds. |
| `fieldValidationError` | A error message that can be set if a metadata-field did not validate. |
| `schemaValidationError` | A error message that corresponds to what have failed during validation against a metadata-schema. |
| `originalFilename` | The original file name of the essence. |
| `originalAudioCodec` | The original audio codec of the essence. |
| `originalVideoCodec` | The original video codec of the essence. |
| `originalHeight` | The original height of the essence. |
| `originalWidth` | The original width of the essence. |
| `originalFormat` | The original container format of the essence. |
| `durationSeconds` | The duration of the item expressed in seconds. |
| `durationTimeCode` | The duration of the item expressed as a time code. |

Deprecated since version 4.14: The system field `user` is not available any more, use the transient field `__user` instead. A transient field `user` ensures backwards compatibility for existing applications but will be removed in a future version.

#### Transient metadata

Transient metadata is a special type of metadata that is not revision controlled and only continuously updated by the system. It can be used to create complex search queries. All transient metadata are prefixed by double underscores.

| Name | Description | Indexed |
|---|---|---|
| __user | The name of the user that imported the item. | Yes |
| __collection | The id of the collection an item belongs to. | Yes |
| __collection_size | The number of collections that the item belongs to. | Yes |
| __ancestor_collection | The id of an ancestor collection of an item. | Yes |
| __ancestor_collection_size | The number of ancestor collections of an item. | Yes |
| __shape | The id of a shape that belongs to the item. | Yes |
| __shape_size | The number of shapes that the item has. | Yes |
| __shape_last_added | The creation date of the newest shape of the item. | Yes |
| __shapetag_{tag}_hash[_{a}] | The checksum of a file of the item, where *a* is the algorithm. | Yes |
| __placeholder_shape_size | The number of placeholder shapes that the item has. | Yes |
| __version | The essence version numbers. | Yes |
| __version_count | The number of essence versions that the item has. | Yes |
| __storage | The id of a storage that has files that belongs to the item. | Yes |
| __storage_size | The number of storages that has files that belongs to the item. | Yes |
| __storage_{tag} | The id of a storage that has files that belongs to the item. | Yes |
| __storage_{tag}_size | The number of storages that has files that belongs to the item. | Yes |
| __storagegroup | The id of a group that has files that belongs to the item. | Yes |
| __storagegroup_size | The number of groups that has files that belongs to the item. | Yes |
| __sequence | The format of a sequence that belongs to the item. | Yes |
| __sequence_size | The number of sequences that the item has. | Yes |
| __metadata_last_modified | The time of the last metadata update. | Yes |
| __external_id | The external identifier assigned to the item. | Yes |
| __deletion_lock_id | The id of the effective deletion lock. | Yes |
| __deletion_lock_expiry | The expiration time of the effective deletion lock. | Yes |

For collections, the following transient metadata fields exist:

| Name | Description | Indexed |
|---|---|---|
| __user | The name of the user that imported the collection. | Yes |
| __child_collection | The id of the collection that the collection contains. | Yes |
| __child_collection_size | The number of collections that the collection contains. | Yes |
| __parent_collection | The id of the collection that the collection belongs to. | Yes |
| __parent_collection_size | The number of collections that the collection belongs to. | Yes |
| __ancestor_collection | The id of an ancestor collection of a collection. | Yes |
| __items_size | The number of direct item children. | Yes |
| __ancestor_collection_size | The number of ancestor collections of a collection. | Yes |
| __metadata_last_modified | The time of the last metadata update. | Yes |
| __folder_mapped | True if the collection maps to a folder, else false. | Yes |
| __child_folder_collection | The id of the folder collection that the collection contains. | Yes |
| __parent_folder_collection | The id of the folder collection that the collection belongs to. | Yes |
| __external_id | The external identifier assigned to the collection. | Yes |
| __deletion_lock_id | The id of the effective deletion lock. | Yes |
| __deletion_lock_expiry | The expiration time of the effective deletion lock. | Yes |

Changed in version 4.15: The deletion lock transient metadata fields were added.

The transient field __user has been introduced and replaces the system field user. A transient field user ensures backwards compatibility for existing applications but will be removed in a future version.

#### File metadata

Metadata can be parsed from some file formats. The metadata is inserted as non-temporal metadata contained in different groups, depending on the source of the metadata. The exact structure of the groups may differ based on the encountered metadata. The parsing of file metadata must be enabled in the *configuration*.

| Name | Type | Description |
|---|---|---|
| `xmp_root` | Group | The root group containing all XMP metadata. |
| `document_root` | Group | The root group for document metadata present in Office and PDF files. |
| `document_text` | Field | The text present in the document |
| `document_metadata` | Group | The group containing the metadata of the document. |

## 2.4 Searching for items (and collections)

### 2.4.1 Searching in Vidispine

Searching in Vidispine is implemented using either Solr or Elasticsearch as the backend. This allows functionality such as boolean operators, faceted searching, term highlighting, search term suggestions, etc. It is possible to search for just items, just collections, or both at the same time, depending on which RESTful resource the search request is made to (`/item`, `/collection` or `/search`). The search criteria are expressed using an XML or JSON document of type ItemSearchDocument, described in more detail below.

---

**Tip:** For best performance

- Don't retrieve the hit count if you don't use it.
- Use *filters* if possible as these can be cached separately and do not affect the score nor highlighting.
- Disable *full text indexing* for fields that contain JSON or other content that should not be included in the full text index.
- Only fetch the specific metadata fields and groups that you need instead of fetching all metadata. See *Get information*.
- If you only want to search in the generic metadata, or if your application does not use timed metadata, then make sure to specify `<intervals>generic</intervals>`.

---

### 2.4.2 Search history

Vidispine stores the search document, as well as the timestamp and user for all searches that are made. If the same user makes an identical search twice, only one entry will be shown in the search history, with the timestamp of the last search.

### 2.4.3 Queries

#### Boolean operators

Boolean operators AND, OR and NOT can be used in search queries. A boolean operator can contain zero or more field-value pairs and zero or more boolean operators.

#### Implicit operators

If no operators are specified operators are implicitly added using the following rules:

### Multiple values within a field

If a field contains multiple values, an implicit OR operator is added.

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>originalFormat</name>
    <value>dv</value>
    <value>mp4</value>
  </field>
</ItemSearchDocument>
```

is logically equivalent to

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <operator operation="OR">
    <field>
      <name>originalFormat</name>
      <value>dv</value>
    </field>
    <field>
      <name>originalFormat</name>
      <value>mp4</value>
    </field>
  </operator>
</ItemSearchDocument>
```

### Multiple field elements at top level

If a document has multiple field elements at top level, an implicit AND operator is added.

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>originalFormat</name>
    <value>dv</value>
  </field>
  <field>
    <name>originalFormat</name>
    <value>mp4</value>
  </field>
</ItemSearchDocument>
```

is logically equivalent to

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <operator operation="AND">
    <field>
      <name>originalFormat</name>
      <value>dv</value>
    </field>
    <field>
      <name>originalFormat</name>
      <value>mp4</value>
    </field>
  </operator>
</ItemSearchDocument>
```

### Text elements

In version 1 of the query syntax text elements are added with an implicit AND operator, and in version 2 it's an implicit OR.

### Example

Searching for items that were not created within the last week and have either the formats `mp4` or `dv`.

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <operator operation="AND">
    <operator operation="NOT">
      <field>
        <name>created</name>
        <range>
          <value>NOW-7DAYS</value>
          <value>NOW</value>
        </range>
      </field>
    </operator>
    <field>
      <name>originalFormat</name>
      <value>mp4</value>
      <value>dv</value>
    </field>
  </operator>
</ItemSearchDocument>
```

### Phrase search

Vidispine supports *wildcard* search and *phrase* search for field type `string` and `string-exact`. A phrase is a group of words surrounded by double quotes, such as "foo bar".

### Wildcard search

The wildcard special character in Vidispine is *, meaning matching zero or more sequential characters.

| | |
|---|---|
| he* | words start with"he", like he, hey, hello |
| h*e | will match he, hope, house, etc. |
| *he | words end with "he", like he, the. |

Note: wildcard in a phrase search is not supported (e.g. `"foo b*"` won't be able to find `foo bar`).

### Range search

Use a range query to find fields with values within a certain range. The minimum and maximum values for the data type of the field can be specified using the attributes `minimum` and `maximum`.

To search for any values:

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>app_score</name>
    <range>
      <value minimum="true"/>
      <value maximum="true"/>
    </range>
  </field>
</ItemSearchDocument>
```

To search for values in range [10..20], that is, inclusive of 10 and 20:

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>app_score</name>
    <range>
      <value>10</value>
      <value>20</value>
    </range>
  </field>
</ItemSearchDocument>
```

The `exclusiveMinimum` and/or the `exclusiveMaximum` attributes can be used to perform exclusive range queries.

For example, to search for values between 10 and 20 (exclusive):

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>app_score</name>
    <range exclusiveMinimum="true" exclusiveMaximum="true">
      <value>10</value>
      <value>20</value>
    </range>
  </field>
</ItemSearchDocument>
```

### Search intervals

By setting <*intervals*> in the *ItemSearchDocument*, search criteria can be applied to metadata within different ranges accordingly:

| | |
|---|---|
| generic | only search generic metadata, a.k.a metadata inside (-INF, +INF) |
| timed | search metadata within ranges other than (-INF, +INF) |
| all | search metadata both *timed* and *generic* metadata (default option) |

For example, search items with only **timed** metadata containing `originalFormat=dv`:

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>originalFormat</name>
    <value>dv</value>
  </field>
  <intervals>timed</intervals>
</ItemSearchDocument>
```

### Group search

Searching items by its metadata groups are supported.

### Example

To find items with any groups:

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <group>
```

```
  </group>
</ItemSearchDocument>
```

To find items without any groups:

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <operator operation="NOT">
    <group>
    </group>
  </operator>
</ItemSearchDocument>
```

To find items without a “movie_info” group:

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <operator operation="NOT">
    <group>
      <name>movie_info</name>
    </group>
  </operator>
</ItemSearchDocument>
```

To find items with a “movie_info” group containing two fields with specific values. Note that the AND is implicit.

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <group>
    <name>movie_info</name>
    <field>
      <name>movie_name</name>
      <value>StarWars</value>
    </field>
    <field>
      <name>episode_no</name>
      <value>1</value>
    </field>
  </group>
</ItemSearchDocument>
```

To find items with a “movie_info” group with an episode number of either 1 or 2.

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <group>
    <name>movie_info</name>
    <operator operation="OR">
      <field>
        <name>episode_no</name>
        <value>1</value>
      </field>
      <field>
        <name>episode_no</name>
        <value>2</value>
      </field>
    </operator>
  </group>
</ItemSearchDocument>
```

To find items with either a “movie_info” or a “video_info” group.

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <operator operation="OR">
      <group>
        <name>movie_info</name>
      </group>
      <group>
        <name>video_info</name>
      </group>
  </operator>
</ItemSearchDocument>
```

### Query syntax versions

In 4.2.0 a new query syntax was introduced. In order to use the new syntax, set the `version` attribute in the search document to `2`. If no version is set, the old query syntax will be used (version 1).

### Version 1

1. The search value of a `string-exact` field is always interpreted literally.
2. The search value of a `string` field is interpreted literally only if it's surrounded by quotation marks. In other cases, implicit `OR` s are used in between the words.
3. Multiple values means OR. Multiple `text` elements means AND.
4. The `noescape` attribute is needed, if user want to search quotation marks or wildcard characters literally in a `string` field;

```
<?xml version="1.0"?>
<ItemSearchDocument>
  <field>
    <name></name>
    <value noescape="true">\"foo bar\"</value>
    <value noescape="true">foo\*</value>
  </field>
</ItemSearchDocument>
```

### Version 2

1. One or more `SPACE` characters means logical AND. So `<value>foo bar</value>` and `<value>foo bar</value>` means searching a field value containing both `foo` and `bar`.
2. Multiple values means OR. To search for `foo` or `bar`, in the title or text:

```
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>title</name>
    <value>foo</value>
    <value>bar</value>
  </field>
</ItemSearchDocument>
```

```
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <text>foo</text>
  <text>bar</text>
</ItemSearchDocument>
```

3. Special characters in Vidispine search are `"`, `SPACE`, `\`, and `*`. Any character followed by `\` is considered as literal. so `\*` means literal `*`, and `\f` is the same as the single character `f`.
4. The characters inside quotes are consider as literal, except `"`. A `\"` is still needed to represent a literal quote inside quotes.
5. The `noescape` attribute of a metadata field value has been removed since Vidispine 4.2.

#### Operators and special characters

To highlight the differences between the two versions:

| Query | Version 1 | Version 2 |
|---|---|---|
| <text>foo bar</text> | foo OR bar | foo AND bar |
| <field>foo bar</field> | foo OR bar | foo AND bar |
| <text>foo</text><br><text>bar</text> | foo AND bar | foo OR bar |
| <field>foo</field><br><field>bar</field> | foo OR bar [1] | foo OR bar |
| “foo bar” | “foo bar” | “foo bar” |
| \”foo\” | \\”foo\\” [2] | \”foo\” |
| foo* | foo* | foo* |
| foo* | foo* [2] | foo* |
| foo_bar [3] | foo\\ OR bar [2] | foo_bar |

#### String types

An example of the differences when searching string fields, assuming a field value of `foo bar`.

| | `foo bar` | | | |
|---|---|---|---|---|
| | Version 1 | | Version 2 | |
| | string | string-exact | string | string-exact |
| `foo` | Y | N | Y | N |
| `FOO` | Y | N | Y | N |
| `foo bar` | Y | Y | Y | N [5] |
| `"foo bar"` | Y | N [4] | Y | Y [6] |
| `foo\ bar` | Y | N | Y | Y [5] |
| `"foo xy"` | N | N | N | N |
| `foo xy` | Y | N | N [7] | N |

### 2.4.4 Filters

A search filter is a query does not affect scoring nor highlighting, similar to a filter query in Solr. A filter can:

- Contain both fields and operators.
- Be named and *excluded from facets*.

[1] Use `<operator operation="AND">` to search for `foo AND bar` for example.
[2] Use `noescape=true` to search for literal `"`, `*` and `SPACE`.
[3] Here _ means SPACE.
[5] `SPACE` is a special character and needs to be escaped in order to get literally meaning.
[4] The character `"` is interpreted literally.
[6] It’s a phrase search, and “string-exact” only have one token in the index, which the same as the query in this case.
[7] It’s `foo OR xy` in version 1, and `foo AND xy` in version 2.

#### Example

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <filter operation="OR" name="productType">
     <field>
       <name>type</name>
       <value>pc</value>
     </field>
     <field>
       <name>type</name>
       <value>phone</value>
     </field>
   </filter>
</ItemSearchDocument>
```

### 2.4.5 Joins

Joint searches on metadata of item, share and file are supported. The old search schema is extended with three new search criterion types: `<item>`, `<shape>`, and `<file>`. Please refer to *xmlSchema.xsd* for the full schema.

Depending on the search result needed(items, shapes, or files), `ItemSearchDocument`, `ShapeSearchDocument` or `FileSearchDocument` should be sent to Vidispine respectively. Those three search documents use the same syntax, only the document names are different.

Note:

1. A `version = 2` document is needed in order to perform the joint search.
2. The `<intervals>` constrain only works for item metadata in a `ItemSearchDocument`. It has not effect in `ShapeSearchDocument` and `FileSearchDocument`.

#### Examples

##### Joins on item search

Find items containing shapes with metadata `shapeCodec=mp4`:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <shape>
    <field>
      <name>shapeCodec</name>
      <value>mp4</value>
    </field>
  </shape>
</ItemSearchDocument>
```

Find items that have generic metadata `title = vidispine`, and contain a shape with `shapeCodec=mp4`, and contain a file with metadata `filetitle = demo`:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <item>
    <field>
      <name>title</name>
      <value>vidispine</value>
    </field>
  </item>
  <shape>
```

```
    <field>
      <name>shapeCodec</name>
      <value>mp4</value>
    </field>
  </shape>
  <file>
    <field>
      <name>filetitle</name>
      <value>demo</value>
    </field>
  </file>
  <intervals>generic</intervals>
</ItemSearchDocument>
```

Find items that have generic metadata `title = vidispine`, and contain a shape with `shapeCodec=mp4`, and contain a file with metadata `filetitle = demo`:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <item>
    <field>
      <name>title</name>
      <value>vidispine</value>
    </field>
  </item>
  <shape>
    <field>
      <name>shapeCodec</name>
      <value>mp4</value>
    </field>
  </shape>
  <file>
    <field>
      <name>filetitle</name>
      <value>demo</value>
    </field>
  </file>
  <intervals>generic</intervals>
</ItemSearchDocument>
```

Find items that have metadata `title = vidispine`, and contain a shape with `shapeCodec=mp4`; the shape must contain a file with metadata `filetitle = demo`:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <item>
    <field>
      <name>title</name>
      <value>item</value>
    </field>
  </item>
  <shape>
    <field>
      <name>shapeCodec</name>
      <value>mp4</value>
    </field>
    <file>
      <field>
```

```
        <name>filetitle</name>
        <value>demo</value>
      </field>
    </file>
  </shape>
</ItemSearchDocument>
```

Operators are also supported as part of a search criterion.

Find items that have metadata `title = vidispine`, or items that have metadata `title = demo` and contain shapes with `shapeCodec=mp4`:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <operator operation="OR">
    <item>
      <field>
        <name>title</name>
        <value>vidispine</value>
      </field>
    </item>
    <operator operation="AND">
      <item>
        <field>
          <name>title</name>
          <value>demo</value>
        </field>
      </item>
      <shape>
        <field>
          <name>shapeCodec</name>
          <value>mp4</value>
        </field>
      </shape>
    </operator>
  </operator>
  <intervals>all</intervals>
</ItemSearchDocument>
```

### Joins on search shapes

Find shapes belong to items that have metadata `title = vidispine`, and the shape should have a file with metadata `filetitle = demo`:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ShapeSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <item>
    <field>
      <name>title</name>
      <value>vidispine</value>
    </field>
  </item>
  <shape>
    <field>
      <name>shapeCodec</name>
      <value>mp4</value>
    </field>
```

```
  </shape>
  <file>
    <field>
      <name>filetitle</name>
      <value>demo</value>
    </field>
  </file>
</ShapeSearchDocument>
```

Find shapes belong to items that have files with metadata `filetitle = demo`, and metadata `title = vidispine`:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ShapeSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <item>
    <field>
      <name>title</name>
      <value>vidispine</value>
    </field>
    <file>
      <field>
        <name>filetitle</name>
        <value>demo</value>
      </field>
    </file>
  </item>
  <shape>
    <field>
      <name>shapeCodec</name>
      <value>mp4</value>
    </field>
  </shape>
</ShapeSearchDocument>
```

### Joins on file search

Find files belong to items with metadata `title=demo`; it should also belongs to shapes with metadata `shape_title=shape`:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<FileSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <item>
    <field>
      <name>title</name>
      <value>demo</value>
    </field>
  </item>
  <shape>
    <field>
      <name>shape_title</name>
      <value>shape</value>
    </field>
  </shape>
</FileSearchDocument>
```

### Joins on collection search

**Note:** Not yet supported with Elasticsearch.

Find collections that have metadata `title=vidispine` or collections contains an item with metadata `title=demo`:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <operator operation="OR">
    <field>
      <name>title</name>
      <value>vidispine</value>
    </field>
    <item>
      <field>
        <name>title</name>
        <value>demo</value>
      </field>
    </item>
  </operator>
</ItemSearchDocument>
```

To find items with specific shapes or files, use a `shape` or `file` query as a subquery of the `item` query.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>title</name>
    <value>vidispine</value>
  </field>
  <item>
    <shape>
      <field>
        <name>shape_title</name>
        <value>demo</value>
      </field>
    </shape>
  </item>
</ItemSearchDocument>
```

**Important:** Using an item subquery is only possible when the search interval is either `generic` or `all`. When using `timed` then no item subquery is allowed.

Find empty collections.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <operator operation="NOT">
    <item>
    </item>
  </operator>
</ItemSearchDocument>
```

### Collection hierarchy joins

New in version 5.5.

Collection joins can be used to search for collections and items based on their relationships by using the `<collection>` element in the ItemSearchDocument. The `<collection>` element may only be used when searching explicitly for items or collections. To specify the relationship between the entities use the `relation` attribute with one of the attributes `child`, `parent`, `ancestor` or `descendant`. If no relation is specified the `child` relation is assumed.

**Note:** Not supported with Elasticsearch.

**Important:** Using a collection subquery is only possible when the search interval is either `generic` or `all`. When using `timed` no collection subquery is allowed.

### Example

Find collections that have a child collection with `title=vidispine`.

```
PUT /collection
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <collection>
    <field>
      <name>title</name>
      <value>vidispine</value>
    </field>
  </collection>
</ItemSearchDocument>
```

### Example

Find collections that have a parent collection with `title=collection1` or `title=collection2` and a descendant collection containing an item with `title=vidispine`.

```
PUT /collection
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <collection relation="parent">
    <operator operation="OR">
      <field>
        <name>title</name>
        <value>collection1</value>
      </field>
      <field>
        <name>title</name>
        <value>collection2</value>
      </field>
    </operator>
  </collection>
```

```
  <collection relation="descendant">
    <item>
      <field>
        <name>title</name>
        <value>vidispine</value>
      </field>
    </item>
  </collection>
</ItemSearchDocument>
```

#### Example

The `<collection>` element may also be nested to search for complex relationships. Find collections that have a grandparent with `title=vidispine`.

```
PUT /collection
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <collection relation="parent">
    <collection relation="parent">
      <field>
        <name>title</name>
        <value>vidispine</value>
      </field>
    </collection>
  </collection>
</ItemSearchDocument>
```

#### Example

The `<collection>` subquery may also be used when searching for items. But only the parent and ancestor relations can be used (items can not contain collections). Find all items that have an ancestor collection with `title=vidispine`.

```
PUT /item
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <collection relation="ancestor">
    <field>
      <name>title</name>
      <value>vidispine</value>
    </field>
  </collection>
</ItemSearchDocument>
```

### 2.4.6 Highlighting

Highlighting can be enabled to determine which part of the metadata that matched the query.

Use the `field` element to enable highlighting for a certain set of fields only.

```
<highlight>
   <field>title</field>
</highlight>
```

### Example

```
PUT /item
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <field>
      <name>title</name> <!-- Search for the words "interview" or "credits" within␣
↪the title -->
      <value>interview</value>
      <value>credits</value>
   </field>
   <highlight> <!-- Having a highlight element will enable highlighting even if it is␣
↪empty -->
      <matchingOnly>true</matchingOnly> <!-- Only highlight fields that actually␣
↪matched the query. -->
      <prefix>[</prefix> <!-- A string that appears before the highlighted text -->
      <suffix>]</suffix> <!-- A string that appears after the highlighted text -->
   </highlight>
</ItemSearchDocument>
```

```
<ItemListDocument>
   <item id="VX-123" start="-INF" end="+INF"> <!-- Matches in the document were on␣
↪the interval [-INF, +INF] -->
      <timespan start="-INF" end="100">  <!-- One match on [-INF, 100] -->
         <field>title</field>
         <value>[Interview] with the CEO.</value> <!-- The word "interview" is␣
↪highlighted with the suffix and prefix -->
      </timespan>
      <timespan start="400" end="+INF"> <!-- Another match on [400, +INF] -->
         <field>title</field>
         <value>Closing [credits]</value> <!-- The word "credits" is highlighted with␣
↪the suffix and prefix -->
      </timespan>
   </item>
</ItemListDocument>
```

**Note:** When searching with cursor in non-generic timespans, only one timespan per item or collection contains highlighting information.

## 2.4.7 Sorting

Results can be sorted using *sortable fields*. Multiple fields can be used for sorting and are used in the order they are given.

It is also possible to sort by relevance by specifying `_relevance` as the field name.

Specify `_type` to sort by type. The type of an item is `item` and `collection` for collections, so if you want collections first in the results, then sort on `_type` in ascending order.

Any field can be used for sorting, it does not need to flagged as sortable. If a field contains multiple values: ascending

order will compare with its minimum value and descending order will compare with its maximum value.

#### Example

Listing all items sorted according to length in descending order and format in ascending order.

```
PUT /item
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <sort>
      <field>durationSeconds</field>
      <order>descending</order>
   </sort>
   <sort>
      <field>originalFormat</field>
      <order>ascending</order>
   </sort>
</ItemSearchDocument>
```

#### Case-insensitive sorting

New in version 5.2.2.

Field values can be sorted case-insensitively if `caseSensitiveSorting=false` is configured in the metadata field definition:

```
<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>string-exact</type>
  <caseSensitiveSorting>false</caseSensitiveSorting>
</MetadataFieldDocument>
```

A reindex is required after the configuration change for it to take effect.

### 2.4.8 Faceting

Faceting is used to show number of matching items for one or more sub-constraints for a given result-set. You might for example be interested in displaying how many of the items returned from a search are of type `video`, how many are of type `audio`, and how many are of type `data`.

There are two types of operations that can be performed, counting and specifying ranges. Counting means that it will count the occurrences of each unique value. When specifying ranges, the number of occurrences within a certain range is counted. Both the start and the end of a range are inclusive and "*" can be used to represent minimum or maximum. Note that faceted search only can be used over non-timed metadata.

### Example

| item | category | price |
|---|---|---|
| VX-251 | tv | 100 |
| VX-252 | radio | 200 |
| VX-253 | tv | 300 |
| VX-254 | phone | 400 |
| VX-255 | radio | 500 |
| VX-256 | radio | 100 |
| VX-257 | phone | 200 |
| VX-258 | phone | 300 |
| VX-259 | phone | 200 |
| VX-260 | phone | 300 |

Consider the items in the table above, together with their metadata on the fields **my_category** and **my_price**. A faceted search that should count the occurrences of each category and the occurrences of prices within the ranges [*, 199], [200, 399] and [400, *] might look like this:

```
PUT /item
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <facet count="false">
        <field>my_price</field>
        <range start="*" end="199"/>
        <range start="200" end="399"/>
        <range start="400" end="*"/>
   </facet>

   <facet count="true">
        <field>my_category</field>
   </facet>
</ItemSearchDocument>
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <hits>13</hits>
   <item id="VX-248" start="-INF" end="+INF"/>
   <item id="VX-249" start="-INF" end="+INF"/>
   <item id="VX-250" start="-INF" end="+INF"/>
   <item id="VX-251" start="-INF" end="+INF"/>
   <item id="VX-252" start="-INF" end="+INF"/>
   <item id="VX-253" start="-INF" end="+INF"/>
   <item id="VX-254" start="-INF" end="+INF"/>
   <item id="VX-255" start="-INF" end="+INF"/>
   <item id="VX-256" start="-INF" end="+INF"/>
   <item id="VX-257" start="-INF" end="+INF"/>
   <item id="VX-258" start="-INF" end="+INF"/>
   <item id="VX-259" start="-INF" end="+INF"/>
   <item id="VX-260" start="-INF" end="+INF"/>
   <facet>
      <field>my_category</field>
      <count fieldValue="phone">5</count>
      <count fieldValue="radio">3</count>
      <count fieldValue="tv">2</count>
   </facet>
   <facet>
      <field>my_price</field>
```

```
      <range start="*" end="199">2</range>
      <range start="200" end="399">6</range>
      <range start="400" end="*">2</range>
   </facet>
</ItemListDocument>
```

Now assume we want to see how the prices are distributed for phones, we could filter the search in the following manner:

```
PUT /item
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <filter>
      <field>
        <name>my_category</name>
        <value>phone</value>
      </field>
   </filter>
   <facet count="false">
        <field>my_price</field>
        <range start="*" end="199"/>
        <range start="200" end="399"/>
        <range start="400" end="*"/>
   </facet>
</ItemSearchDocument>
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <hits>5</hits>
   <item id="VX-254" start="-INF" end="+INF"/>
   <item id="VX-257" start="-INF" end="+INF"/>
   <item id="VX-258" start="-INF" end="+INF"/>
   <item id="VX-259" start="-INF" end="+INF"/>
   <item id="VX-260" start="-INF" end="+INF"/>
   <facet>
      <field>my_price</field>
      <range start="*" end="199">0</range>
      <range start="200" end="399">4</range>
      <range start="400" end="*">1</range>
   </facet>
</ItemListDocument>
```

The opposite is also possible, to see the distribution of the categories over a range of prices.

```
PUT /item
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">

   <filter>
        <field>
          <name>my_price</name>
          <range start="200" end="399"/>
        </field>
   </filter>

   <facet count="true">
```

```
        <field>my_category</field>
   </facet>
</ItemSearchDocument>
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <hits>6</hits>
   <item id="VX-252" start="-INF" end="+INF"/>
   <item id="VX-253" start="-INF" end="+INF"/>
   <item id="VX-257" start="-INF" end="+INF"/>
   <item id="VX-258" start="-INF" end="+INF"/>
   <item id="VX-259" start="-INF" end="+INF"/>
   <item id="VX-260" start="-INF" end="+INF"/>
   <facet>
      <field>my_category</field>
      <count fieldValue="phone">4</count>
      <count fieldValue="radio">1</count>
      <count fieldValue="tv">1</count>
   </facet>
</ItemListDocument>
```

For counted facets, it is possible to supply a `minCount`, thereby excluding any fields that has a count lower than the specified minimum count.

```
PUT /item
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <facet count="true" minCount="3">
        <field>my_category</field>
   </facet>
</ItemSearchDocument>
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <hits>13</hits>
   <item id="VX-248" start="-INF" end="+INF"/>
   <item id="VX-249" start="-INF" end="+INF"/>
   <item id="VX-250" start="-INF" end="+INF"/>
   <item id="VX-251" start="-INF" end="+INF"/>
   <item id="VX-252" start="-INF" end="+INF"/>
   <item id="VX-253" start="-INF" end="+INF"/>
   <item id="VX-254" start="-INF" end="+INF"/>
   <item id="VX-255" start="-INF" end="+INF"/>
   <item id="VX-256" start="-INF" end="+INF"/>
   <item id="VX-257" start="-INF" end="+INF"/>
   <item id="VX-258" start="-INF" end="+INF"/>
   <item id="VX-259" start="-INF" end="+INF"/>
   <item id="VX-260" start="-INF" end="+INF"/>
   <facet>
      <field>my_category</field>
      <count fieldValue="phone">5</count>
      <count fieldValue="radio">3</count>
   </facet>
</ItemListDocument>
```

By default, at most 100 facet counts will be returned. By using the `maxResults` attribute, this behaviour can be changed.

```
PUT /item
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <facet count="true" maxResults="1">
        <field>my_category</field>
   </facet>
</ItemSearchDocument>
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <hits>13</hits>
   <item id="VX-248" start="-INF" end="+INF"/>
   <item id="VX-249" start="-INF" end="+INF"/>
   <item id="VX-250" start="-INF" end="+INF"/>
   <item id="VX-251" start="-INF" end="+INF"/>
   <item id="VX-252" start="-INF" end="+INF"/>
   <item id="VX-253" start="-INF" end="+INF"/>
   <item id="VX-254" start="-INF" end="+INF"/>
   <item id="VX-255" start="-INF" end="+INF"/>
   <item id="VX-256" start="-INF" end="+INF"/>
   <item id="VX-257" start="-INF" end="+INF"/>
   <item id="VX-258" start="-INF" end="+INF"/>
   <item id="VX-259" start="-INF" end="+INF"/>
   <item id="VX-260" start="-INF" end="+INF"/>
   <facet>
      <field>my_category</field>
      <count fieldValue="phone">5</count>
   </facet>
</ItemListDocument>
```

### Facet exclusion

One or more search filters can be excluded from a facet using `<exclude>` tags. Facets can be named to make it possible to distinguish between different facets, for example when using multiple facets on the same field but with different excludes.

The facet exclusion is similar to how one can tag and exclude filters in Solr (https://wiki.apache.org/solr/SimpleFacetParameters#Tagging_and_excluding_Filters).

### Example

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <filter name = "tvFilter">
     <field>
        <name>my_category</name>
        <value>tv</value>
     </field>
   </filter>
   <filter name = "priceFilter">
      <field>
        <name>my_price</name>
        <range start="200" end="399"/>
      </field>
   </filter>

   <facet count="true">
        <field>my_category</field>
```

```
   </facet>

   <facet name="excludeTv" count="true">
        <field>my_category</field>
        <exclude>tvFilter</exclude>
        <!-- <exclude>tvFilter2</exclude> Multiple exclusions -->
   </facet>
</ItemSearchDocument>
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item id="VX-253" start="-INF" end="+INF"/>
   <facet>
      <field>my_category</field>
      <count fieldValue="tv">1</count>
      <count fieldValue="phone">0</count>
      <count fieldValue="radio">0</count>
   </facet>
   <facet name="excludeTv">
      <field>my_category</field>
      <count fieldValue="tv">4</count>
      <count fieldValue="phone">1</count>
      <count fieldValue="radio">1</count>
   </facet>
</ItemListDocument>
```

### 2.4.9 Spell checking

Search terms can be checked against a dictionary. This enables "Did you mean..." types of searches. The dictionary used is built from the search index and updated periodically.

#### Example

Consider a user is intending to searching for the "original duration" but misspells both words:

```
PUT /item
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <text>orignal durraton</text>

    <suggestion> <!-- Enables spell checking -->
        <maximumSuggestions>2</maximumSuggestions> <!-- Optional: Specifies the␣
↪maximum number of suggestions -->
        <accuracy>0.7</accuracy> <!-- Optional: A value between 0.0 (least accurate)␣
↪and 1.0 (most accurate) of how accurate the spell check should be -->
    </suggestion>
</ItemSearchDocument>
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <hits>0</hits>
  <suggestion>
    <term>orignal</term> <!-- A misspelled search term -->

    <!-- A list of suggestions, with the most likely suggestion being first -->
    <suggestion>original</suggestion>
    <suggestion>ordinal</suggestion>
  </suggestion>
```

```
  <suggestion>
    <term>durraton</term>
    <suggestion>duration</suggestion>
  </suggestion>
</ItemListDocument>
```

### 2.4.10 Autocompletion

Text can be autocompleted against the search index.

#### Example

Assuming the user intends to type "original duration". The user first starts typing "original":

```
PUT /search/autocomplete
Content-Type: application/xml

<AutocompleteRequestDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <text>orig</text>
  <maximumSuggestions>3</maximumSuggestions>
</AutocompleteRequestDocument>
```

```
<AutocompleteResponseDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <suggestion>original</suggestion>
  <suggestion>origin</suggestion>
  <suggestion>originated</suggestion>
</AutocompleteResponseDocument>
```

Then the user continues to start typing "duration":

```
<AutocompleteRequestDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <text>original dur</text>
  <maximumSuggestions>3</maximumSuggestions>
</AutocompleteRequestDocument>
```

```
<AutocompleteResponseDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <suggestion>original duration</suggestion>
</AutocompleteResponseDocument>
```

#### Autocomplete on metadata fields

You can also autocomplete on specific metadata fields. In order to make the autocompletion case insensitive, the metadata field should be set as `<index>extend</index>`.

#### Example:

A metadata field `foo_bar` with config:

```
<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
         <type>string-exact</type>
         <index>extend</index>
</MetadataFieldDocument>
```

and this filed contains multiple values: "Animal", "Sky", "Animal and Sky", "animal and sky"

An auto-complete request with user input "animal a":

```
PUT /search/autocomplete
Content-Type: application/xml

<AutocompleteRequestDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>foo_bar</field>
  <text>animal a</text>
</AutocompleteRequestDocument>
```

will give result:

```
<AutocompleteResponseDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <suggestion>Animal and Sky</suggestion>
  <suggestion>animal and sky</suggestion>
</AutocompleteResponseDocument>
```

### Autocomplete within a search

It is also possible to get autocomplete suggestions while searching. The suggestions will then only match against the search result set.

### Example

Let's say you have a large number of assets, all of which are tagged with one or more tags. A user might want to filter down the result set, so in this example we search for any assets with category "stock_photo", and at the same time we request autocomplete suggestions for the tag field matching the string "hi".

```
PUT /item
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>my_category</name>
    <value>stock_photo</value>
  </field>
  <autocomplete>
    <field>my_tag</field>
    <text>hi</text>
  </autocomplete>
</ItemSearchDocument>
```

The result might then look like:

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <hits>5</hits>
  <item id="VX-6934" start="-INF" end="+INF">
    <timespan start="-INF" end="+INF"/>
  </item>
  <item id="VX-3464" start="-INF" end="+INF">
    <timespan start="-INF" end="+INF"/>
  </item>
  <item id="VX-2658" start="-INF" end="+INF">
    <timespan start="-INF" end="+INF"/>
  </item>
  <item id="VX-7234" start="-INF" end="+INF">
    <timespan start="-INF" end="+INF"/>
  </item>
  <item id="VX-3723" start="-INF" end="+INF">
```

```
    <timespan start="-INF" end="+INF"/>
  </item>
  <autocomplete>
    <field>my_tag</field>
    <suggestion>highres</suggestion>
    <suggestion>hills</suggestion>
    <suggestion>history</suggestion>
  </autocomplete>
</ItemListDocument>
```

### 2.4.11 Search Boost

New in version 4.16.

It is also possible to boost some field values during a search.

To enable search boosting, either add the boost factor in the search document or as the default boost factor in the metadata field definition. Also, `_score` has to be used as the sorting field.

#### Example

To find items that have the word `phoenix` in either `title_field` or `description_field`. But the matches in `title_field` are more important.

```
PUT /item
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine" version="2">
  <operator operation="OR">
    <field>
      <name>title_field</name>
      <value boost="10">phoenix</value>
    </field>
    <field>
      <name>description_field</name>
      <value>phoenix</value>
    </field>
  </operator>
  <sort>
    <field>_score</field>
    <order>descending</order>
  </sort>
</ItemSearchDocument>
```

Alternatively, the boost factor can also be set the the metadata field definition.

```
<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <name>title_field</name>
  <type>string</type>
  <boost>10.0</boost>
</MetadataFieldDocument>
```

## 2.5 Caching

### 2.5.1 Search result caching

### Saved searches

When searching using the PUT (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.6) method, the query parameter `save=true` can be used save the query and request parameters for later retrieval. This can be used to improve performance for queries that are performed frequently, as the saved search endpoint supports conditional GET using ETag.

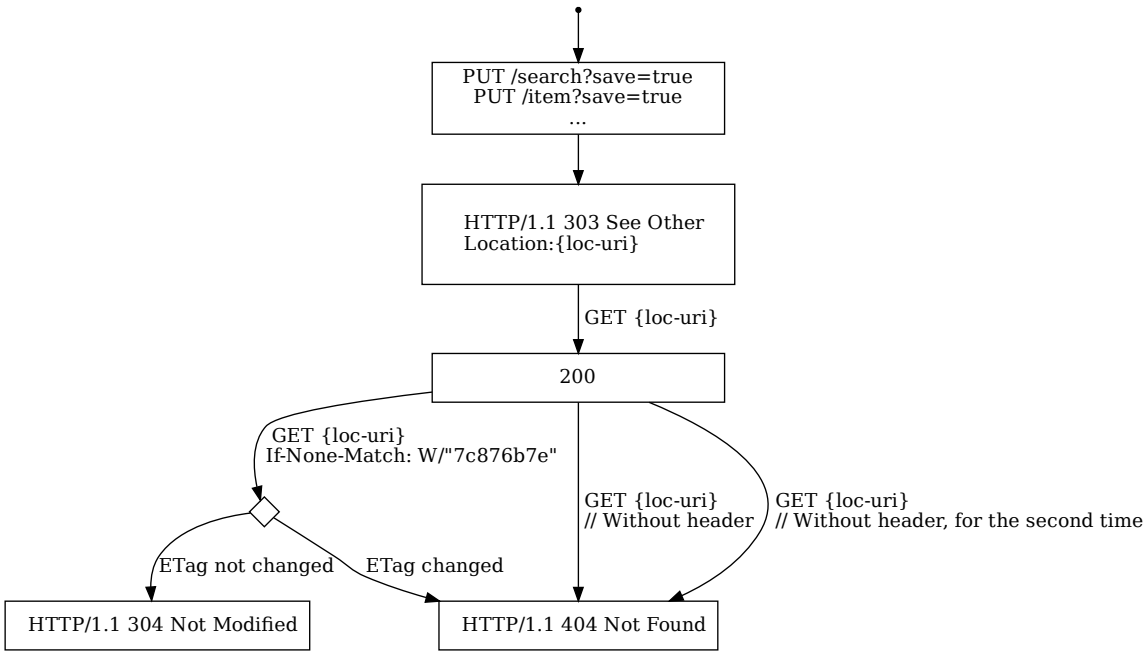
The response will then have status 303 See Other (http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.3.4) and `Location` header, from where the search result can be fetched. The URI from the `Location` header supports the `ETag` and `If-None-Match` headers for GET (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3) requests. GET requests to the location URI without the `If-None-Match` header with return the search result with it's ETag, or `404` if the saved search has been invalidated and removed.

For item or collection search, the saved search will be invalidated and removed if any entity of that type is re-indexed.

---

**Note:** The ETags returned from saved search requests are weak ETags; meaning that the "Content-Type" headers and query parameters won't affect the value of the ETag.

---




### Example

An item search request is first made using `save=true`:

```
PUT API/item?save=true
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <text>squirrel</text>
</ItemSearchDocument>
```

```
HTTP/1.1 303 See Other
Location: http://localhost:8080/API/item/saved/f8297c9d02083d66731b4438415fd26b?
↪type=ITEM
```

Then, to fetch the query results:

```
GET http://localhost:8080/API/item/saved/f8297c9d02083d66731b4438415fd26b?type=ITEM
```

```
HTTP/1.1 200 OK
Content-Type: application/xml
ETag: W/"50521d364314765d9f672279375939b8"

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <hits>1</hits>
  <item id="VX-26424" start="-INF" end="+INF">
    <timespan start="-INF" end="+INF"/>
  </item>
</ItemListDocument>
```

After that, the ETag can be used to perform a conditional GET:

```
GET http://localhost:8080/API/item/saved/f8297c9d02083d66731b4438415fd26b?type=ITEM
If-None-Match: W/"50521d364314765d9f672279375939b8"
```

```
HTTP/1.1 304 Not Modified
```

## 2.6 Metadata projections

Vidispine supports two kinds of conversion tools for automating integration with other systems.

**Projection** A metadata projection is a bidirectional XSLT transformation, meant to simplify integration of the Vidispine system with several third party systems.

**Auto-projection** The auto-projection is used to interact on metadata changes. For example, a change to one field may automatically trigger changes to other fields.

### 2.6.1 Projections

A projection consists of an incoming and an outgoing XSLT transformation.

- The incoming projection transforms information in some format to a format supported by Vidispine.
- The outgoing projection transforms information from Vidispine to a some other format.

When you use projections to transform item metadata then the outgoing projection will transform a MetadataListDocument and the incoming projection must produce a MetadataDocument.

#### Projection id

Projections are identified by a projection id of the regular expression form: `[_A-Za-z][_A-Za-z0-9]*`, maximum 32 characters. The projection is is case sensitive.

#### Example

This is an example of valid XSL:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:
↪vs="http://xml.vidispine.com/schema/vidispine">

<xsl:template match="/">
  <metadata>
  <item><xsl:value-of select="vs:MetadataListDocument/vs:item/@id"/></item>
    <xsl:for-each select="vs:MetadataListDocument/vs:item/vs:metadata/vs:timespan/vs:
↪field">
      <metadataField>
        <name><xsl:value-of select="vs:name"/></name>
        <xsl:for-each select="vs:value">
          <value><xsl:value-of select="."/></value>
        </xsl:for-each>
      </metadataField>
    </xsl:for-each>
  </metadata>
</xsl:template>
</xsl:stylesheet>
```

### 2.6.2 XSLT 2.0

XSL Transformations version 2.0 (http://www.w3.org/TR/xslt20/) are supported. Remember to specify the correct version in the stylesheet.

### 2.6.3 Job Information

It is possible to access job information in the XSLT. This is done by adding the element `<vs:vsXSLTVersion>2</vs:vsXSLTVersion>(xmlns:vs="http://xml.vidispine.com/schema/vidispine'` at the global level of the XSLT. When the `xsltVersion` option is set, the actual input to the transformation is no longer a `MetadataListDocument`, but an `ExportInformationDocument`. The new input contains two element:

**`metadataList`** The same as the old input to the transformation.

**`job`** The current job, as outputted by `/API/job/{jobid}?metadata=true`.

#### Example

The following example uses both XSLT v2.0 and the job information:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:
↪vs="http://xml.vidispine.com/schema/vidispine">
    <vs:vsXSLTVersion>2</vs:vsXSLTVersion>
    <xsl:template match="/">
        <root>
            <job>
                <xsl:value-of select="vs:ExportInformationDocument/vs:job/vs:jobId/
↪text()"/>
            </job>
            <custom>
                <xsl:for-each select="vs:ExportInformationDocument/vs:job/vs:data[vs:
↪key='custom']/vs:value/tokenize(.,',')">
                    <data>
                        <xsl:value-of select="."/>
                    </data>
```

```
                </xsl:for-each>
            </custom>
        </root>
    </xsl:template>
</xsl:stylesheet>
```

### 2.6.4 Auto-projection rules

The auto projection is used to interact on metadata changes. For example, a change to one field may automatically trigger changes to other fields.

An AutoProjectionRuleDocument contains of four parts: `<step>`,`<description>`,`<inputFilters>` and `<triggers>`.

#### MetadataWrapperDocument

During the projection, a temporary structure called "MetadataWrapperDocument" is created for manipulation.

A MetadataWrapperDocument contains some of the fields below, depending on the values of `inputFilters` in AutoProjectionRuleDocument :

| | |
|---|---|
| `metadata` | The new incoming item metadata |
| `oldMetadata` | The old metadata of the item |
| `shapeMetadata` | The new incoming shape metadata |
| `shape` | The shape list of the item |
| `bulkyMetadata` | The new incoming bulky metadata of the item or shape |
| `oldBulkyMetadata` | The old bulky metadata of the item |

#### Projection steps

Multiple projection steps can be defined in different `<step>`, with their execution order, description, and script/XSLT respectively. Please note that `<script>` and `<xslt>` are used to hold JavaScript and XSLT respectively and each step can only contain one of them.

Example:

```
<step>
  <order>1</order>
  <description>step1 description</description>
  <script>...</script>
</step>
<step>
  <order>2</order>
  <description>step2 description</description>
  <xslt>...</xslt>
</step>
```

#### Input filters

Input filter defines which information should goes into the MetadataWrapperDocument during the projection. There are two kinds of filters: `inputFilter` and `bulkyMetadataKeysRegex`

Legal values of `inputFilter` are

| | |
|---|---|
| `oldMetadata` | Add old metadata of the item into the MetadataWrapperDocument |
| `shapeDocument` | Add old shape metadata of the item into the MetadataWrapperDocument |

All bulky metadata of the item whose key matches the pattern defined in `bulkyMetadataKeysRegex` will go into MetadataWrapperDocument. Multiple filters are allowed.

Example:

```
<inputFilters>
  <inputFilter>oldMetadata</inputFilter>
  <inputFilter>shapeDocument</inputFilter>
  <bulkyMetadataKeysRegex>.*</bulkyMetadataKeysRegex>
</inputFilters>
```

#### Rule triggers

Rule triggers defines what kinds of metadata update triggers this rule. They are:

| | |
|---|---|
| `itemMetadata` | Rule triggered if there is an item metadata update |
| `shapeMetadata` | Rule triggered if there is a shape metadata update |
| `bulkyMetadata` | Rule triggered if there is a bulky metadata update |

Multiple triggers are allowed.

```
<triggers>
  <trigger>itemMetadata</trigger>
  <trigger>shapeMetadata</trigger>
</triggers>
```

### 2.6.5 Auto-projection using JavaScript

A JavaScript projection is created by including the JavaScript in the `script` element of AutoProjectionRuleDocument. The script will be interpreted using a *JavaScript engine*.

A number of global variables are defined for the script to use:

- api
- helper
- wrapper

#### The api object

Please see *The api object*.

#### The helper object

Please see *The metadatahelper object*. In Auto-projection scripts, it is also available under the name `helper`.

#### The wrapper object

The wrapper object represents the MetadataWrapperDocument during the projection. Below are available functions:

`wrapper.`**`getMetadata`**`()`
: Get the new incoming item metadata.

  **Returns** *MetadataType*.

`wrapper.`**`getOldMetadata`**`()`
: Get the old metadata of the item.

  **Returns** *MetadataListType*.

`wrapper.`**`getShapeMetadata`**`()`
Get the new incoming shape metadata.

**Returns** *SimpleMetadataType*.

`wrapper.`**`getShape`**`()`
Get the shape list of the item.

**Returns** List<*ShapeType*>.

`wrapper.`**`getBulkyMetadata`**`()`
Get the new incoming bulky metadata of the item/shape.

**Returns** *BulkyMetadataType*.

`wrapper.`**`getOldBulkyMetadata`**`()`
Get the old bulky metadata of the item.

**Returns** *BulkyMetadataType*.

`wrapper.`**`getOldBulkyMetadata`**`()`
Get the old bulky metadata of the item.

**Returns** *BulkyMetadataType*.

`wrapper.`**`setMetadata`**`(`*value*`)`
Assign a new metadata to the wrapper document.

**Arguments**

- **`value`** – *MetadataType*

There are two ways to apply projection results to an item:

1. If the rule is triggered by an item metadata update, one should manipulate the object reference returned by `wrapper.getMetadata()` directly. Because Vidispine will take that object as the projection result.
2. If the rule is triggered by a shape metadata update or bulky metadata update, one should use the `api` object to update the item metadata:

```
var metadata = helper.createMetadata();
...
var xml = helper.metadataToStr(metadata);
var id = wrapper.getTargetId();
var result = api.path("item/" + id + "/metadata").input(xml, "application/xml").
↪put();
```

### 2.6.6 Auto-projection using XSLT

A XSLT projection is created by including the XSL script in the `xsl` element of AutoProjectionRuleDocument.

The transformation result could either be a MetadataDocument or MetadataWrapperDocument. If the result is a MetadataWrapperDocument, the value of `metadata` element will be used as the projection result.

During a shape/bulky metadata update, one need to set up another step using the JavaScript `api` object to update the item metadata.

Example:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:
↪vs="http://xml.vidispine.com/schema/vidispine">
    <xsl:template match="/">
        <MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
            <timespan start="-INF" end="+INF">
```

```
                <xsl:for-each select="vs:MetadataWrapperDocument/vs:metadata/vs:
↪timespan/vs:field">
                    <field>
                        <name>
                            <xsl:value-of select="vs:name"/>
                        </name>
                        <value><xsl:value-of select="vs:value"/>+projection</value>
                    </field>
                </xsl:for-each>
            </timespan>
        </MetadataDocument>
    </xsl:template>
</xsl:stylesheet>
```

## 2.7 Metadata migrations

Vidispine has support for migrating metadata to adhere to a new structure. For example, you might have changed the group hierarchies in your metadata schema, and want to migrate old items and collections to the new schema. This is done by posting a migration definition. Vidispine will then automatically go through all the metadata in the system and migrate it.

### 2.7.1 Migration operations

There are a number of operations available for metadata migrations:

- **Move** This is used to move a field or a group from one position in the hierarchy to another.
- **Rename** This can be used to rename fields. Note that the new name must already be defined as a metadata field in the system, and the data types of the old and new fields must be compatible (e.g. a string field cannot be renamed to a date field, since it could cause invalid values to be introduced)
- **Delete** Used to delete a field or a group from a metadata hierarchy.

### 2.7.2 Migration definition

Migrations are defined using XML (or JSON). Here is an example of a migration containing all of the above operations:

```
<MetadataSchemaMigrationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <move type="field">
    <from>
      <group>
        <name>Film</name>
        <field>
          <name>actor</name>
        </field>
      </group>
    </from>
    <to>
      <group>
        <name>Film</name>
        <group>
          <name>Personnel</name>
        </group>
      </group>
    </to>
  </move>
  <rename>
```

```
    <from>
      <group>
        <name>Film</name>
          <field>
            <name>internal_title</name>
          </field>
      </group>
    </from>
    <to>production_id</to>
  </rename>
  <delete type="group">
    <target>
      <group>
        <name>Film</name>
        <group>
          <name>Soundtrack</name>
        </group>
      </group>
    </target>
  </delete>
</MetadataSchemaMigrationDocument>
```

The above migration would perform three operations:

- A move operation on any `actor` field that is located in the `Timespan > Film` group. It would instead be placed in `Timespan > Film > Personnel` group.
- A rename operation. It would rename any `internal_title` field located in the `Timespan > Film` group. It would rename it to `production_id`.
- A delete operation which would delete any group matching `Timespan -> Film -> Soundtrack`.

## 2.8 Metadata datasets

A metadata dataset is a set of metadata values that have semantic relations between each other. Datasets can be used to validate metadata documents.

### 2.8.1 Defining the dataset

A dataset is defined using a RDF (https://www.w3schools.com/xml/xml_rdf.asp) document. Vidispine supports creating a dataset using either a RDF/XML (https://www.w3.org/TR/rdf-syntax-grammar/) document or a TURTLE (https://www.w3.org/TR/turtle/) document.

For example, the geographical hierarchy relations of `USA`, `New York`, `California`, `Los Angeles` and `San Francisco` can be defined as follows:

```
@prefix r: <http://example.com/random/id#> .
@prefix c: <http://example.com/country/#> .
@prefix st: <http://example.com/state/#> .
@prefix city: <http://example.com/city/#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .

c:usa   skos:definition  "country" ;
        skos:member      r:bid1 ;
        skos:prefLabel   "USA" .

r:bid1  a       rdf:Bag ;
```

```
        rdf:_1  st:ny ;
        rdf:_2  st:ca .

st:ny   skos:definition  "state" ;
        skos:prefLabel   "New York" .

st:ca   skos:definition  "state" ;
        skos:member      r:bid2 ;
        skos:prefLabel   "California" .

r:bid2  a       rdf:Bag ;
        rdf:_1  c:la ;
        rdf:_2  c:sf .

c:la    skos:definition  "city" ;
        skos:prefLabel   "Los Angeles" .

c:sf    skos:definition  "city" ;
        skos:prefLabel   "San Francisco" .
```

In the above dataset, five subjects (or resources) have been defined: `USA`, `New York`, `California`, `Los Angeles` and `San Francisco`. Each subject has its own id (`c:usa`, `st:ca` etc.), and two predicates (or properties) `skos:prefLabel` and `skos:definition`; representing the subject's "display value" and "hierarchical level" respectively. The hierarchical relationships between the subjects are defined by the `skos:member` and the RDF container `rdf:Bag`.

You can also use self-defined vocabularies. The example below uses self-defined `hasState` and `hasCity` properties to represent the geographical relationship:

```
@prefix c: <http://example.com/country/#> .
@prefix st: <http://example.com/state/#> .
@prefix city: <http://example.com/city/#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .

c:usa   skos:definition  "country" ;
        skos:prefLabel   "USA" ;
        c:hasState  st:ca , st:ny .

st:ca   st:hasCity   city:la , city:sf ;
        skos:definition  "state" ;
        skos:prefLabel   "California" .

st:ny   skos:definition  "state" ;
        skos:prefLabel   "New York" .

city:sf skos:definition  "city" ;
        skos:prefLabel   "San Francisco" .

city:la skos:definition  "city" ;
        skos:prefLabel   "Los Angeles" .
```

### 2.8.2 Create the dataset

The dataset above can be used to *create a metadata dataset* in Vidispine:

```
PUT /metadata/dataset/mytestmodel

Content-Type: text/turtle
```

```
@prefix c: <http://example.com/country/#> .
@prefix st: <http://example.com/state/#> .
@prefix city: <http://example.com/city/#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .

c:usa    skos:definition  "country" ;
         skos:prefLabel   "USA" ;
         c:hasState  st:ca , st:ny .

st:ca    st:hasCity   city:la , city:sf ;
         skos:definition  "state" ;
         skos:prefLabel   "California" .

st:ny    skos:definition  "state" ;
         skos:prefLabel   "New York" .

city:sf skos:definition  "city" ;
         skos:prefLabel   "San Francisco" .

city:la skos:definition  "city" ;
         skos:prefLabel   "Los Angeles" .
```

If the display values of a dataset model is changed, make sure to reindex any entities that have metadata set on them using these fields.

### 2.8.3 Configure metadata fields

After creating the dataset, the metadata fields needs to be configured accordingly:

```
<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <name>rdf_country</name>
  <type>string</type>
  <constraint>
    <dataset>mytestmodel</dataset>
    <levelProperty>skos:definition</levelProperty>
    <levelValue>country</levelValue>
    <value>skos:prefLabel</value>
  </constraint>
</MetadataFieldDocument>
```

```
<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <name>rdf_state</name>
  <type>string</type>
  <constraint>
    <dataset>mytestmodel</dataset>
    <levelProperty>skos:definition</levelProperty>
    <levelValue>state</levelValue>
    <value>skos:prefLabel</value>
    <parent>rdf_country</parent>
  </constraint>
</MetadataFieldDocument>
```

```
<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <name>rdf_city</name>
  <type>string</type>
  <constraint>
    <dataset>mytestmodel</dataset>
    <levelProperty>skos:definition</levelProperty>
    <levelValue>city</levelValue>
    <value>skos:prefLabel</value>
    <parent>rdf_state</parent>
  </constraint>
</MetadataFieldDocument>
```

The configuration above defines three metadata fields: `rdf_country`, `rdf_state` and `rdf_city`, whose values are restricted by the metadata dataset `mytestmodel`.

- `<dataset>`: Which dataset the field value should be validated against.
- `<levelProperty>`: Should be the property in the dataset that defines a level value.
- `<levelValue>`: Which level in the dataset does this metadata field belong to.
- `<value>`: The display value of the metadata field.
- `<parent>`: (Optional). The parent field if any. This defines the field hierarchy. Fields in the same hierarchy will be validated together.
- `<validationGroup>`: (Optional). Containing an **ordered**, comma separated value, defining which fields should be validated together, and the **hierarchical (validation) order** of those fields.

Changed in version 4.17.2: The `parent` element was added. The `validationGroup` element was deprecated.

### 2.8.4 Updating metadata

There are two ways to post metadata documents containing semantically related fields:

1. Using the value directly, like we have always been doing:

```
<?xml version="1.0" encoding="UTF-8"?>
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    <field>
      <name>rdf_country</name>
      <value>USA</value>
    </field>
    <field>
      <name>rdf_state</name>
      <value>New York</value>
    </field>
  </timespan>
</MetadataDocument>
```

2. Using the corresponding subject id from the dataset:

```
<?xml version="1.0" encoding="UTF-8"?>
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    <field>
      <name>rdf_country</name>
      <value id="c:usa"/>
```

```
      <!-- or the full URI -->
      <!-- <value id="http://example.com/country#usa"/> -->
    </field>
    <field>
      <name>rdf_state</name>
      <value id="st:ny"/>
    </field>
  </timespan>
</MetadataDocument>
```

The resulting value will contain both `id` and the "display value":

```
GET item/(item-id)/metadata
```

```
...
<field uuid="783a6bc1-7917-4aa4-9d37-0c1dd4f6787f">
    <name>rdf_country</name>
    <value id="c:usa" uuid="459b3d37-c3a1-4ae6-8ad7-ab4a934f3a42">USA</value>
</field>
<field uuid="41ffab92-e984-4a36-b8ed-2aae41be56e6">
    <name>rdf_state</name>
    <value id="st:ny" uuid="898cb68c-b661-4923-8ded-3e9cb02a200b">New York</value>
</field>
...
```

The `includeConstraintValue` query parameter can be used to only fetch the "display value" of the specified fields:

```
GET item/(item-id)/metadata?includeConstraintValue=rdf_country
```

```
...
<field uuid="783a6bc1-7917-4aa4-9d37-0c1dd4f6787f">
    <name>rdf_country</name>
    <value id="c:usa" uuid="459b3d37-c3a1-4ae6-8ad7-ab4a934f3a42">USA</value>
</field>
<field uuid="41ffab92-e984-4a36-b8ed-2aae41be56e6">
    <name>rdf_state</name>
    <value id="st:ny" uuid="898cb68c-b661-4923-8ded-3e9cb02a200b"/>
</field>
...
```

### 2.8.5 Searching for dataset values

New in version 4.17.3.

Searching for entities with metadata from a dataset can be done via regular search. See *Search*

For example:

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>rdf_state</name>
    <value>New York</value>
  </field>
</ItemSearchDocument>
```

Or:

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <text>New York</text>
</ItemSearchDocument>
```

### 2.8.6 Validation of metadata values

Vidispine will try to validate the incoming metadata document accordingly to the constraint configured on the metadata fields. Fields defined in the same hierarchy and that belongs to the same timespan and metadata group will be validated together.

For example, this is an invalid document because `London` is not a city in `USA` according to the metadata dataset:

```
<?xml version="1.0" encoding="UTF-8"?>
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    <field>
      <name>rdf_city</name>
      <value>London</value>
    </field>
    <field>
      <name>rdf_country</name>
      <value>USA</value>
    </field>
  </timespan>
</MetadataDocument>
```

This is an invalid document because the field values in `my_test_group` are not correct:

```
<?xml version="1.0" encoding="UTF-8"?>
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    <field>
      <name>rdf_city</name>
      <value>Los Angeles</value>
    </field>
    <field>
      <name>rdf_state</name>
      <value>California</value>
    </field>
    <group>
      <name>my_test_group</name>
      <field>
        <name>rdf_city</name>
        <value>Los Angeles</value>
      </field>
      <field>
        <name>rdf_state</name>
        <value>New York</value>
      </field>
    </group>
  </timespan>
</MetadataDocument>
```

This is a valid document, because `rdf_state2` belongs to a different hierarchy than `rdf_city` and `rdf_state` belong to. And they all contain valid values:

```
<?xml version="1.0" encoding="UTF-8"?>
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    <field>
      <name>rdf_city</name>
      <value>Los Angeles</value>
    </field>
    <field>
      <name>rdf_state</name>
      <value>California</value>
    </field>
    <field>
      <name>rdf_state2</name>
      <value>New York</value>
    </field>
  </timespan>
</MetadataDocument>
```

### 2.8.7 Retrieving allowed values

To get all allowed value of a metadata field:

```
GET metadata-field/rdf_city/allowed-values
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ConstraintValueListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <value id="city:man">Manchester</value>
  <value id="city:nyc">New York City</value>
  <value id="city:la">Los Angeles</value>
  <value id="city:buf">Buffalo</value>
  <value id="city:sf">San Francisco</value>
  <value id="city:roc">Rochester</value>
  <value id="city:lnd">London</value>
</ConstraintValueListDocument>
```

To find all allowed values:

```
POST metadata-field/rdf_city/allowed-values
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MetadataFieldValueConstraintListDocument xmlns="http://xml.vidispine.com/schema/
↪vidispine">
  <constraint>
    <field>rdf_country</field>
    <value>USA</value>
    <!-- or use the constraint subject id -->
    <id>http://example.com/country#usa</id>
  </constraint>
  <constraint>
    <field>rdf_state</field>
    <value>New York</value>
  </constraint>
</MetadataFieldValueConstraintListDocument>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ConstraintValueListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <value id="city:nyc">New York City</value>
</ConstraintValueListDocument>
```

## 2.9 Subtitles

Vidispine supports adding subtitles to an item. They can then for example be exported to Final Cut. Subtitles can also be used with sequences and can be included in the video when a sequence is rendered.

### 2.9.1 Subtitle metadata fields and groups

To add subtitles to a Vidispine item, the metadata field group `stl_subtitle` must be used. The group should be placed within a `timespan` corresponding to the in- and out timecodes the subtitles should be displayed. Within this group, the following fields can be set:

- `stl_text`. This sets the actual text which should be displayed. Multiple lines are delimited by a line feed character.
- `stl_justification`. Determines the justification of multiple lines of text.

  **`left`** all lines are aligned to left border of text bounding box

  **`center`** all lines are aligned in center of text bounding box

  **`right`** all lines are aligned to right border of text bounding box
- `stl_xrelative`. Horizontal position of base point relative to full video frame.

  **0.0** left border

  **1.0** right border
- `stl_yrelative`. Vertical position of base point relative to full video frame.

  **0.0** top border

  **1.0** bottom border
- `stl_horizontalbase`. Horizontal position of base point relative to text bounding box.

  **0.0 (or `left`)** base point is left border of bounding box.

  **0.5 (or `center`)** base point is center of bounding box.

  **1.0 (or `right`)** base point is right border of of bounding box.
- `stl_verticalbase`. Vertical position of base point relative to text bounding box.

  **0.0 (or `top`)** base point is top border of of bounding box.

  **0.5 (or `middle`)** base point is middle of bounding box.

  **1.0 (or `bottom`)** base point is bottom border of bounding box.
- `stl_sizerelative`. Height of font relative to full video frame.
- `stl_color`. Color of text. Can be standard colors (`red`) or hexadecimal (`#ff0000`).
- `stl_outline`. Type of outline.

  **(none)** no outline

  **`bar`** rectangular outline

**stroke** fat stroke around text

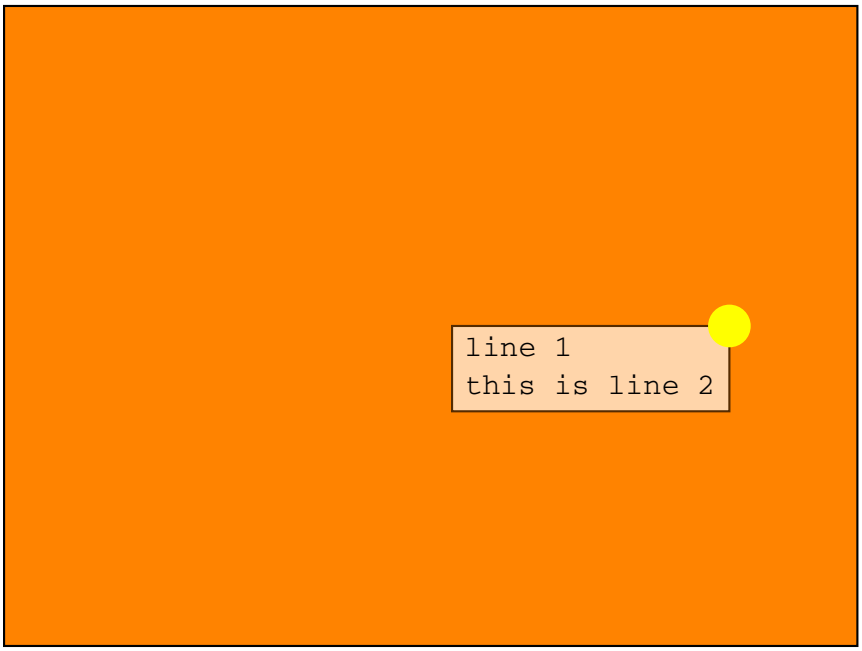
- `stl_outlinecolor`. Color of outline.
- `stl_outlinesize`. Size (margin) of outline.
- `stl_font`. Font of subtitle.

  **monospace** fixed-width font (default)

  **sans** sans-serif font

  **serif** font with serifs

### Example




- `stl_justification=left`
- `stl_xrelative=0.9`
- `stl_yrelative=0.5`
- `stl_horizontalbase=right`
- `stl_verticalbase=top`

The subtitle language can be extracted from the `.stl` file itself or set using `jobmetadata`, key `subtitleLanguage`; `jobmetadata` has a higher priority.

### Example

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="10@PAL" end="100@25">
    <group>
      <name>stl_subtitle</name>
      <field><name>stl_justification</name><value>left</value></field>
      <field><name>stl_vertical</name><value>6</value></field>
      <field><name>stl_text</name><value>some text&#13;&#10;actually two lines</value>
↪</field>
```

```
    </group>
  </timespan>
</MetadataDocument>
```

### 2.9.2 Rendering subtitles in a sequence

If you have a sequence attached to an item in Vidispine the subtitle metadata can be included in the output file. To do this, you need to use a shape tag where `<burnSubtitles>true</burnSubtitles>` is set in the `<video>` element. Note that overlapping subtitle timespans are not allowed and will cause the render job to fail.

#### Example

Let's say we have an item `VX-811` which has a sequence attached to it, and the following metadata:

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  ...
  <timespan start="72140@PAL" end="72260@PAL">
    <group>
      <name>stl_subtitle</name>
      <field><name>stl_justification</name><value>center</value></field>
      <field><name>stl_text</name><value>No, I am your father.</value></field>
    </group>
  </timespan>
  <timespan start="72320@PAL" end="72490@PAL">
    <group>
      <name>stl_subtitle</name>
      <field><name>stl_justification</name><value>center</value></field>
      <field><name>stl_text</name><value>No... that's not true!&#13;&#10;That's␣
↪impossible!</value></field>
    </group>
  </timespan>
  ...
</MetadataDocument>
```

And we have the following shape-tag called MP4_512_SUB:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mp4</format>
  <audio>
    <codec>aac</codec>
    <bitrate>96000</bitrate>
  </audio>
  <video>
    <scaling>
      <width>512</width>
      <height>288</height>
    </scaling>
    <codec>h264</codec>
    <bitrate>2000000</bitrate>
    <framerate>
      <numerator>1</numerator>
      <denominator>25</denominator>
    </framerate>
    <burnSubtitles>true</burnSubtitles>
  </video>
</TranscodePresetDocument>
```

Then a render job is started using:

```
POST /item/VX-811/sequence/render?tag=MP4_512_SUB
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine"><jobId>VX-1436</jobId>
↪<user>admin</user><started>2013-03-08T13:02:15.654Z</started><status>READY</status>
↪<type>CONFORM</type><priority>MEDIUM</priority></JobDocument>
```

This will render the sequence and include any subtitle metadata as subtitles in the output video.

### 2.9.3 SCC support

To export subtitle metadata for an item in SCC format, use the *SCC export resource*.

```
GET /item/VX-56/metadata/export/scc
```

```
Scenarist_SCC V1.0

00:00:00:00  942c 942c 9420 9420 9470 9470 54e5 f8f4

00:00:10:00  942f 942f

00:00:20:00  942c 942c
```

### 2.9.4 TTML support

Subtitles for an item can also be retrieved in TTML format using *Export to TTML*.

```
GET /item/{id}/metadata/export/ttml
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<tt xmlns:ns2="http://www.w3.org/ns/ttml#styling" xmlns="http://www.w3.org/ns/ttml"␣
↪xmlns:ns4="http://www.w3.org/ns/ttml#metadata" xmlns:ns3="urn:ebu:tt:style" xmlns:
↪ns5="http://www.w3.org/ns/ttml#parameter" xmlns:ns6="urn:ebu:tt:metadata" ns5:
↪frameRate="25" ns5:cellResolution="50 30" ns2:extent="704px 576px" xml:lang="en">
  <head>
    <metadata>
      <ns6:documentMetadata>
        <ns6:documentTargetAspectRatio>4:3</ns6:documentTargetAspectRatio>
        <ns6:documentTotalNumberOfSubtitles>11</ns6:documentTotalNumberOfSubtitles>
        <ns6:documentMaximumNumberOfDisplayableCharacterInAnyRow>40</ns6:
↪documentMaximumNumberOfDisplayableCharacterInAnyRow>
        <ns6:documentStartOfProgramme>00:00:00:00</ns6:documentStartOfProgramme>
        <ns6:documentCountryOfOrigin>GB</ns6:documentCountryOfOrigin>
        <ns6:documentPublisher>Institut fuer Rundfunktechnik   </ns6:
↪documentPublisher>
      </ns6:documentMetadata>
    </metadata>
    <styling>
      <style xml:id="textCenter" ns2:textAlign="center"/>
      <style xml:id="defaultStyle" ns2:fontFamily="monospaceSansSerif" ns2:fontSize=
↪"1c 1c" ns2:lineHeight="normal" ns2:textAlign="center" ns2:color="white" ns2:
↪backgroundColor="transparent" ns2:fontStyle="normal" ns2:fontWeight="normal" ns2:
↪textDecoration="none"/>
      <style xml:id="whiteOnblackDH" ns2:fontSize="1c 2c" ns2:color="white" ns2:
↪backgroundColor="black"/>
    </styling>
```

```
    <layout>
      <region xml:id="bottom" ns2:origin="10% 10%" ns2:extent="80% 80%" ns2:
↪displayAlign="after" ns2:padding="0c" ns2:writingMode="lrtb"/>
      <region xml:id="top" ns2:origin="10% 10%" ns2:extent="80% 80%" ns2:displayAlign=
↪"before" ns2:padding="0c" ns2:writingMode="lrtb"/>
    </layout>
  </head>
  <body>
    <div xml:id="SGN1" style="defaultStyle">
      <p region="top" style="textCenter" begin="00:00:00:00" end="00:00:02:10">
        <br/>
        <span style="whiteOnblackDH">two-line</span>
        <br/>
        <span style="whiteOnblackDH">top</span>
      </p>
      <p region="top" style="textCenter" begin="00:00:02:14" end="00:00:04:21">
        <br/>
        <span style="whiteOnblackDH">one-line top</span>
      </p>
      ...
    </div>
  </body>
</tt>
```

## 2.10 Examples

### 2.10.1 Creating fields/groups, modifying and moving metadata

Let's say that we have an item that contains a sports game. We want to record the goals that have occurred within the game. To do this we have the triple (time, team, player), where the time is the real-world time when the goal took place, the player that scored and the team the player plays for.

#### Creating the metadata fields

First to create the field for time, we choose the data type "date" since we want it to be indexed, but we will use it as temporal metadata so it is not applicable to be a sortable field.

```
PUT /metadata-field/sport_time
Content-Type: application/xml

<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <type>date</type>
</MetadataFieldDocument>
```

As for creating the team and the player, we use the same reasoning above, with the exception of that we want a string instead

```
PUT /metadata-field/sport_team
Content-Type: application/xml

<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <type>string</type>
</MetadataFieldDocument>
```

```
PUT /metadata-field/sport_player
Content-Type: application/xml
```

```
<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <type>string</type>
</MetadataFieldDocument>
```

### Creating the metadata field group

With the fields created we now want a way to group these fields together so we create a field group called "goal".

```
PUT /metadata-field/field-group/goal
```

Now we simply add the fields, created above, to the group.

```
PUT /metadata-field/field-group/goal/sport_time
```

```
PUT /metadata-field/field-group/goal/sport_team
```

```
PUT /metadata-field/field-group/goal/sport_player
```

Retrieving the group:

```
GET /metadata-field/field-group/goal
```

```
<MetadataFieldListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <field sortable="false">
      <name>sport_time</name>
      <type>date</type>
   </field>
   <field sortable="false">
      <name>sport_player</name>
      <type>string</type>
   </field>
   <field sortable="false">
      <name>sport_team</name>
      <type>string</type>
   </field>
</MetadataFieldListDocument>
```

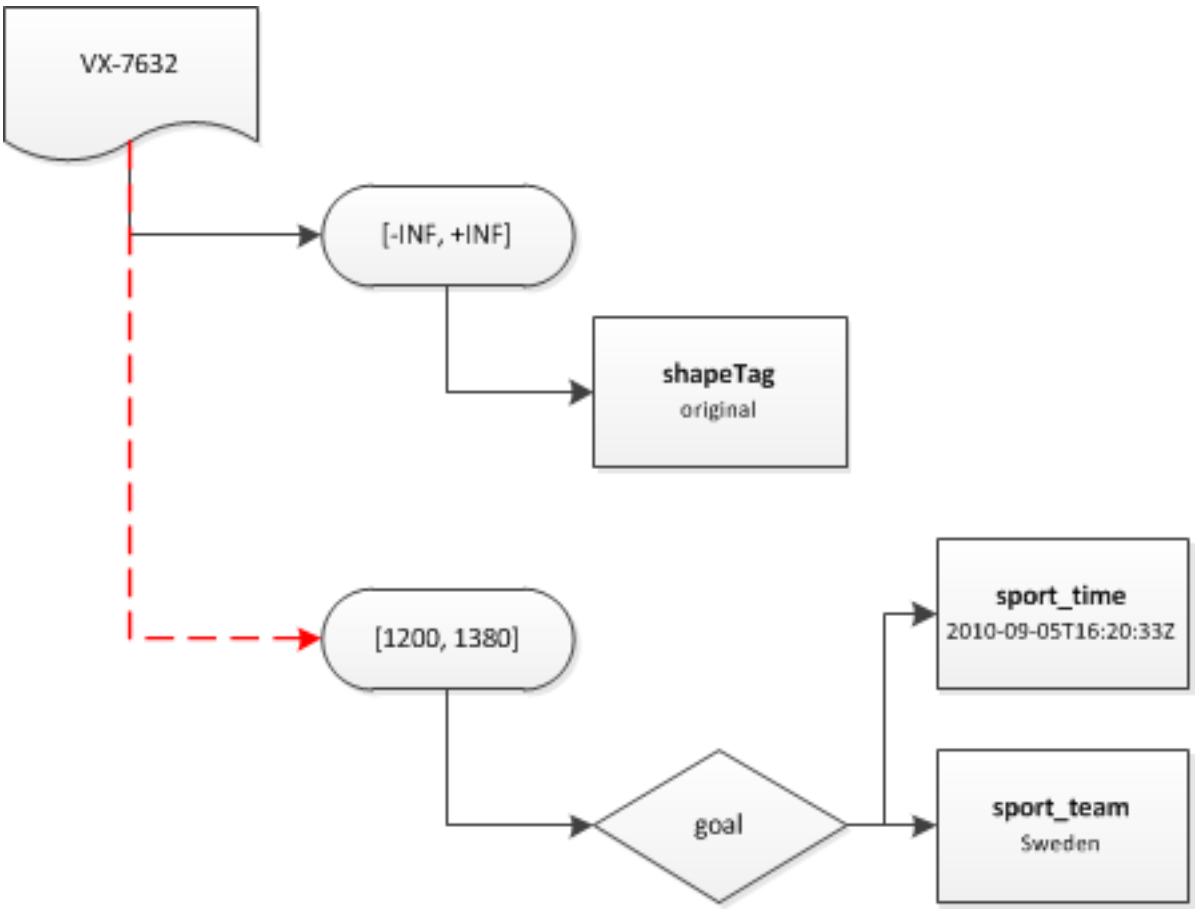
### Modifying metadata

Let's say that the item VX-7632 contains two goals that occurred during a game that matches the triples (time='2010-09-05T16:20:33Z', team='Sweden', player='Pete') and (time='2010-09-05T16:42:05Z', team='Germany', player='Bob'). Within the item the first goal can be seen between the time codes (1200, 1380) and the second goal between the time codes (2700, 2940).

Each step will contain a diagram, where the dashed red line illustrates the semantics of the request being performed.

### Adding the first goal

Adding the first goal without adding the player:



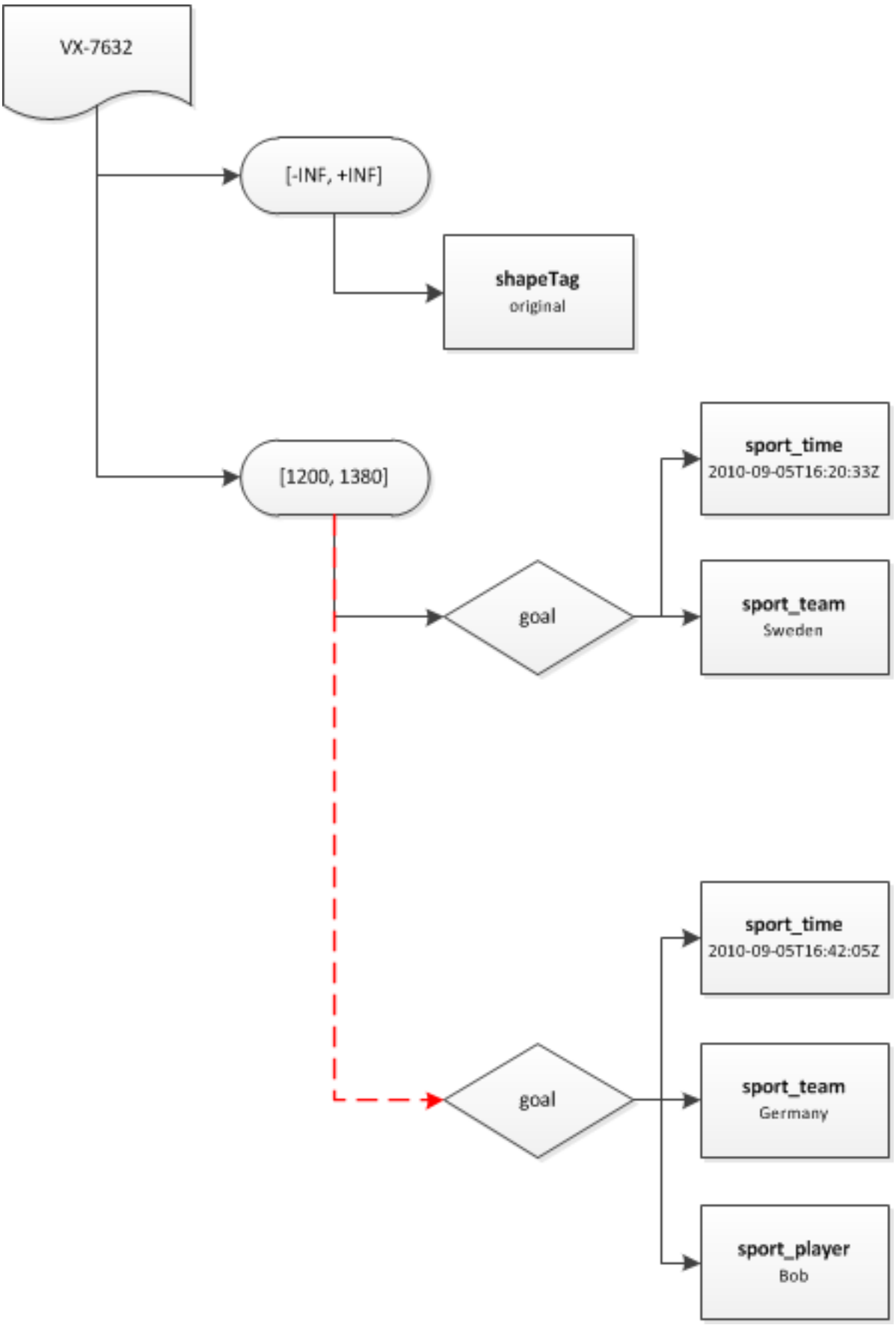

```
PUT /item/VX-7632/metadata
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan start="1200" end="1380">
      <group mode="add">
         <name>goal</name>
         <field>
            <name>sport_time</name>
            <value>2010-09-05T16:20:33Z</value>
         </field>
         <field>
            <name>sport_team</name>
            <value>Sweden</value>
         </field>
      </group>
   </timespan>
</MetadataDocument>
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <item id="VX-7632">
        <metadata>
            <revision>VX-16295,VX-16296,VX-16299</revision>
            <timespan end="1380" start="1200">
                <group change="VX-16299" timestamp="2010-09-08T15:36:01.836+02:00"␣
↪user="admin" uuid="1f89d35d-02b6-4871-aa17-62c5ed4992f4">
```

```
                    <name>goal</name>
                    <field change="VX-16299" timestamp="2010-09-08T15:36:01.836+02:00
↪" user="admin" uuid="915b6023-f374-4432-832c-a2c48c1efb56">
                        <name>sport_time</name>
                        <value change="VX-16299" timestamp="2010-09-08T15:36:01.
↪836+02:00" user="admin" uuid="cce8f89a-a220-4e53-8734-5831a3a4eb77">2010-09-05T16:
↪20:33Z</value>
                    </field>
                    <field change="VX-16299" timestamp="2010-09-08T15:36:01.836+02:00
↪" user="admin" uuid="d9d9b21c-171d-402b-878d-cefa5b3f9727">
                        <name>sport_team</name>
                        <value change="VX-16299" timestamp="2010-09-08T15:36:01.
↪836+02:00" user="admin" uuid="7493df98-be67-4fb6-97fe-a89b9e501207">Sweden</value>
                    </field>
                </group>
            </timespan>
            <timespan end="+INF" start="-INF">
                <field change="VX-16295" timestamp="2010-09-08T11:00:15.833+02:00"␣
↪user="system" uuid="7c5c49f9-c740-4b0a-93e8-81490fb65799">
                    <name>shapeTag</name>
                    <value change="VX-16295" timestamp="2010-09-08T11:00:15.833+02:00
↪" user="system" uuid="9c2945d5-3480-436e-bfbb-2444e586961d">original</value>
                </field>
            </timespan>
        </metadata>
    </item>
</MetadataListDocument>
```

### Adding the second goal

Adding the second goal, accidentally to same timespan as the first goal:


VX-7632
[-INF, +INF]
shapeTag
original
[1200, 1380]
sport_time
2010-09-05T16:20:33Z
goal
sport_team
Sweden
sport_time
2010-09-05T16:42:05Z
goal
sport_team
Germany
sport_player
Bob

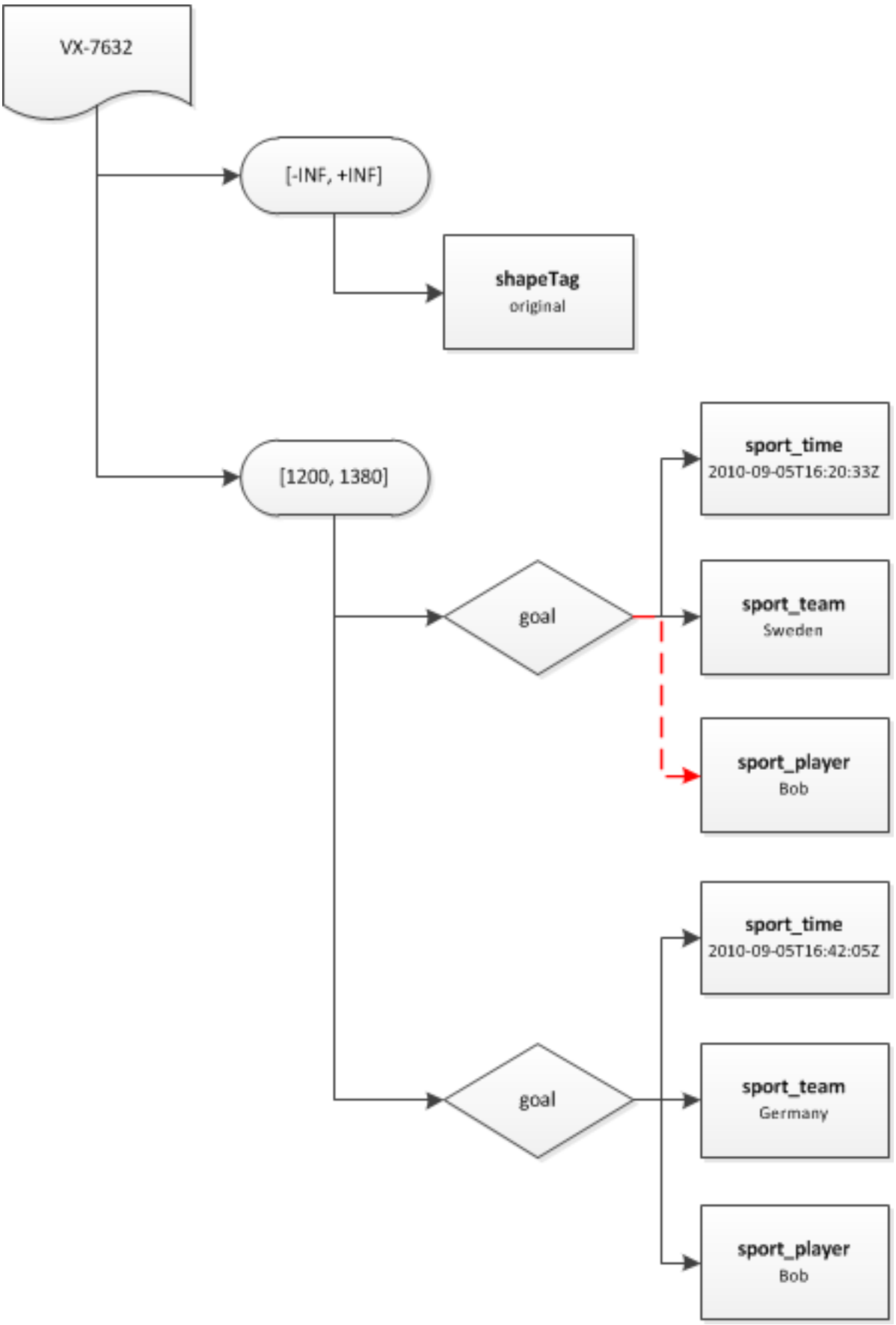
```
PUT /item/VX-7632/metadata
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan start="1200" end="1380">
      <group mode="add">
         <name>goal</name>
         <field>
            <name>sport_time</name>
            <value>2010-09-05T16:42:05Z</value>
         </field>
         <field>
            <name>sport_team</name>
            <value>Germany</value>
         </field>
         <field>
            <name>sport_player</name>
            <value>Bob</value>
         </field>
      </group>
   </timespan>
</MetadataDocument>
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <item>
        <metadata>
            <revision>VX-16295,VX-16296,VX-16299,VX-16300</revision>
            <timespan end="1380" start="1200">
                <group change="VX-16299" timestamp="2010-09-08T15:36:01.836+02:00"␣
↪user="admin" uuid="1f89d35d-02b6-4871-aa17-62c5ed4992f4">
                    <name>goal</name>
                    <field change="VX-16299" timestamp="2010-09-08T15:36:01.836+02:00
↪" user="admin" uuid="915b6023-f374-4432-832c-a2c48c1efb56">
                        <name>sport_time</name>
                        <value change="VX-16299" timestamp="2010-09-08T15:36:01.
↪836+02:00" user="admin" uuid="cce8f89a-a220-4e53-8734-5831a3a4eb77">2010-09-05T16:
↪20:33Z</value>
                    </field>
                    <field change="VX-16299" timestamp="2010-09-08T15:36:01.836+02:00
↪" user="admin" uuid="d9d9b21c-171d-402b-878d-cefa5b3f9727">
                        <name>sport_team</name>
                        <value change="VX-16299" timestamp="2010-09-08T15:36:01.
↪836+02:00" user="admin" uuid="7493df98-be67-4fb6-97fe-a89b9e501207">Sweden</value>
                    </field>
                </group>
                <group change="VX-16300" timestamp="2010-09-08T15:38:28.715+02:00"␣
↪user="admin" uuid="0e9a54eb-0b90-4ed9-ac68-d2cb5d7abc73">
                    <name>goal</name>
                    <field change="VX-16300" timestamp="2010-09-08T15:38:28.715+02:00
↪" user="admin" uuid="4e5ffd77-ab59-46fe-9939-47ab61df7523">
                        <name>sport_team</name>
                        <value change="VX-16300" timestamp="2010-09-08T15:38:28.
↪715+02:00" user="admin" uuid="2a64f141-b3aa-4686-973c-7c254a0b77cb">Germany</value>
                    </field>
                    <field change="VX-16300" timestamp="2010-09-08T15:38:28.715+02:00
↪" user="admin" uuid="2444055e-40e0-49b6-8493-0f68df82f01a">
                        <name>sport_player</name>
                        <value change="VX-16300" timestamp="2010-09-08T15:38:28.
↪715+02:00" user="admin" uuid="441404a4-882c-458a-af88-b2fad592d71c">Bob</value>
```

```
                    </field>
                    <field change="VX-16300" timestamp="2010-09-08T15:38:28.715+02:00
↪" user="admin" uuid="01d54bbb-d01e-4c6f-880e-1c1cbc4e598e">
                        <name>sport_time</name>
                        <value change="VX-16300" timestamp="2010-09-08T15:38:28.
↪715+02:00" user="admin" uuid="71dd57e3-4a39-41ad-b351-78c4bc20ac0b">2010-09-05T16:
↪42:05Z</value>
                    </field>
                </group>
            </timespan>
            <timespan end="+INF" start="-INF">
                <field change="VX-16295" timestamp="2010-09-08T11:00:15.833+02:00"␣
↪user="system" uuid="7c5c49f9-c740-4b0a-93e8-81490fb65799">
                    <name>shapeTag</name>
                    <value change="VX-16295" timestamp="2010-09-08T11:00:15.833+02:00
↪" user="system" uuid="9c2945d5-3480-436e-bfbb-2444e586961d">original</value>
                </field>
            </timespan>
        </metadata>
    </item>
</MetadataListDocument>
```

### Modifying the first goal

Adding the missing player to first goal:


VX-7632
[-INF, +INF]
shapeTag
original
[1200, 1380]
goal
sport_time
2010-09-05T16:20:33Z
sport_team
Sweden
sport_player
Bob
goal
sport_time
2010-09-05T16:42:05Z
sport_team
Germany
sport_player
Bob

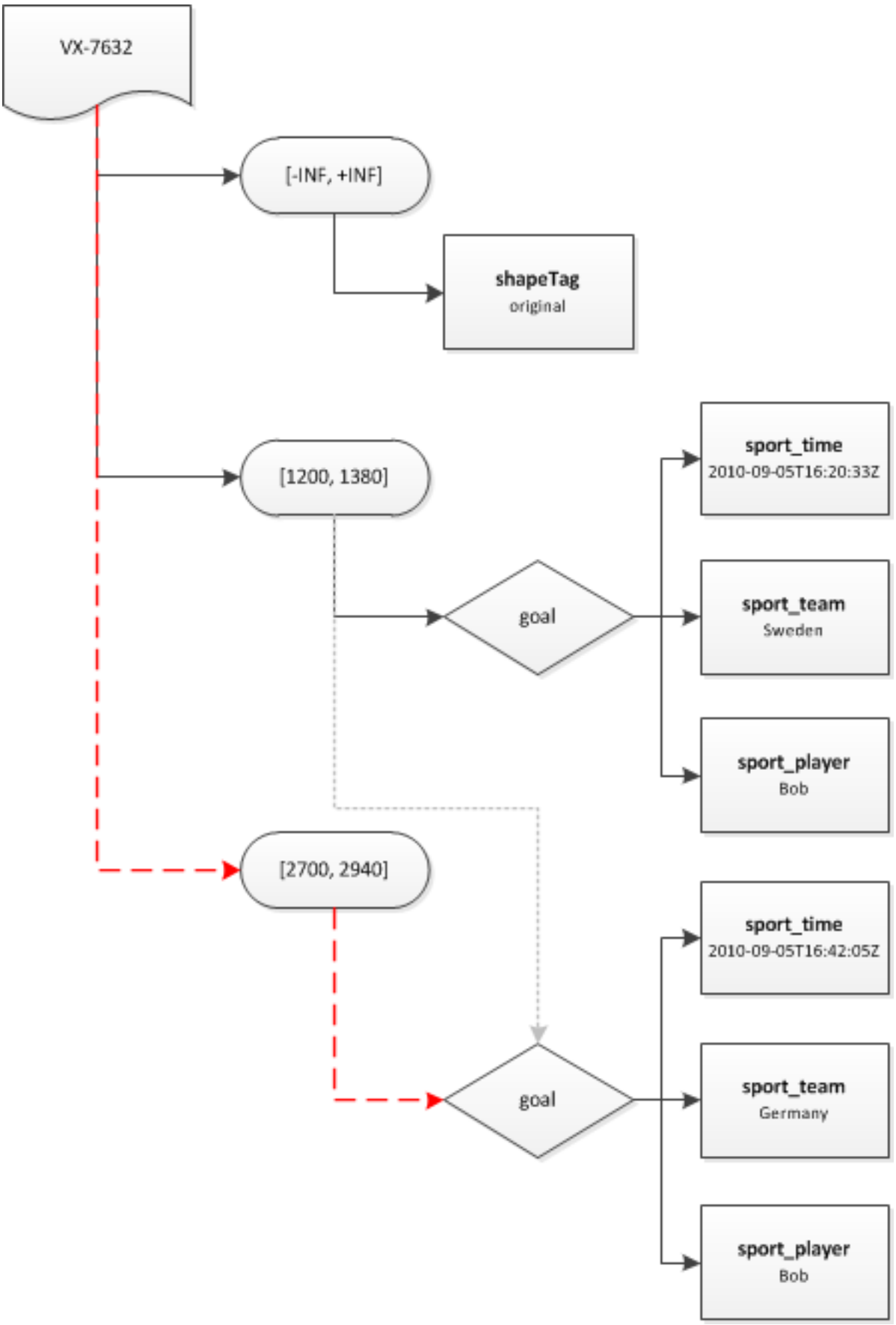
```
PUT /item/VX-7632/metadata
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan start="1200" end="1380">
      <group uuid="1f89d35d-02b6-4871-aa17-62c5ed4992f4">
         <name>goal</name>
         <field mode="add">
            <name>sport_player</name>
            <value>Pete</value>
         </field>
      </group>
   </timespan>
</MetadataDocument>
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <item>
        <metadata>
            <revision>VX-16301,VX-16295,VX-16296,VX-16299,VX-16300</revision>
            <timespan end="1380" start="1200">
                <group change="VX-16301" timestamp="2010-09-08T15:41:22.212+02:00"␣
↪user="admin" uuid="1f89d35d-02b6-4871-aa17-62c5ed4992f4">
                    <name>goal</name>
                    <field change="VX-16301" timestamp="2010-09-08T15:41:22.212+02:00
↪" user="admin" uuid="e374df6f-deb5-4d5e-bfea-d1c2ae6df9aa">
                        <name>sport_player</name>
                        <value change="VX-16301" timestamp="2010-09-08T15:41:22.
↪212+02:00" user="admin" uuid="dbb77bcb-c3e5-4d9e-90a6-3114eca1d091">Pete</value>
                    </field>
                    <field change="VX-16299" timestamp="2010-09-08T15:36:01.836+02:00
↪" user="admin" uuid="915b6023-f374-4432-832c-a2c48c1efb56">
                        <name>sport_time</name>
                        <value change="VX-16299" timestamp="2010-09-08T15:36:01.
↪836+02:00" user="admin" uuid="cce8f89a-a220-4e53-8734-5831a3a4eb77">2010-09-05T16:
↪20:33Z</value>
                    </field>
                    <field change="VX-16299" timestamp="2010-09-08T15:36:01.836+02:00
↪" user="admin" uuid="d9d9b21c-171d-402b-878d-cefa5b3f9727">
                        <name>sport_team</name>
                        <value change="VX-16299" timestamp="2010-09-08T15:36:01.
↪836+02:00" user="admin" uuid="7493df98-be67-4fb6-97fe-a89b9e501207">Sweden</value>
                    </field>
                </group>
                <group change="VX-16300" timestamp="2010-09-08T15:38:28.715+02:00"␣
↪user="admin" uuid="0e9a54eb-0b90-4ed9-ac68-d2cb5d7abc73">
                    <name>goal</name>
                    <field change="VX-16300" timestamp="2010-09-08T15:38:28.715+02:00
↪" user="admin" uuid="4e5ffd77-ab59-46fe-9939-47ab61df7523">
                        <name>sport_team</name>
                        <value change="VX-16300" timestamp="2010-09-08T15:38:28.
↪715+02:00" user="admin" uuid="2a64f141-b3aa-4686-973c-7c254a0b77cb">Germany</value>
                    </field>
                    <field change="VX-16300" timestamp="2010-09-08T15:38:28.715+02:00
↪" user="admin" uuid="2444055e-40e0-49b6-8493-0f68df82f01a">
                        <name>sport_player</name>
                        <value change="VX-16300" timestamp="2010-09-08T15:38:28.
↪715+02:00" user="admin" uuid="441404a4-882c-458a-af88-b2fad592d71c">Bob</value>
                    </field>
```

```
                    <field change="VX-16300" timestamp="2010-09-08T15:38:28.715+02:00
↪" user="admin" uuid="01d54bbb-d01e-4c6f-880e-1c1cbc4e598e">
                        <name>sport_time</name>
                        <value change="VX-16300" timestamp="2010-09-08T15:38:28.
↪715+02:00" user="admin" uuid="71dd57e3-4a39-41ad-b351-78c4bc20ac0b">2010-09-05T16:
↪42:05Z</value>
                    </field>
                </group>
            </timespan>
            <timespan end="+INF" start="-INF">
                <field change="VX-16295" timestamp="2010-09-08T11:00:15.833+02:00"␣
↪user="system" uuid="7c5c49f9-c740-4b0a-93e8-81490fb65799">
                    <name>shapeTag</name>
                    <value change="VX-16295" timestamp="2010-09-08T11:00:15.833+02:00
↪" user="system" uuid="9c2945d5-3480-436e-bfbb-2444e586961d">original</value>
                </field>
            </timespan>
        </metadata>
    </item>
</MetadataListDocument>
```

### Moving metadata

Since the second is placed in the wrong timespan it can be corrected by moving it.


VX-7632
[-INF, +INF]
shapeTag
original
[1200, 1380]
sport_time
2010-09-05T16:20:33Z
goal
sport_team
Sweden
sport_player
Bob
[2700, 2940]
sport_time
2010-09-05T16:42:05Z
goal
sport_team
Germany
sport_player
Bob

```
PUT /item/VX-7632/metadata/move?start=2700&end=2940&uuid=0e9a54eb-0b90-4ed9-ac68-
 ↪d2cb5d7abc73
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <revision>VX-16301,VX-16295,VX-16296,VX-16299,VX-16300</revision>
    <timespan start="1200" end="1380">
        <group uuid="1f89d35d-02b6-4871-aa17-62c5ed4992f4" user="admin" timestamp=
↪"2010-09-08T15:41:22.212+02:00" change="VX-16301">
            <name>goal</name>
            <field uuid="e374df6f-deb5-4d5e-bfea-d1c2ae6df9aa" user="admin" timestamp=
↪"2010-09-08T15:41:22.212+02:00" change="VX-16301">
                <name>sport_player</name>
                <value uuid="dbb77bcb-c3e5-4d9e-90a6-3114eca1d091" user="admin"␣
↪timestamp="2010-09-08T15:41:22.212+02:00" change="VX-16301">Pete</value>
            </field>
            <field uuid="915b6023-f374-4432-832c-a2c48c1efb56" user="admin" timestamp=
↪"2010-09-08T15:36:01.836+02:00" change="VX-16299">
                <name>sport_time</name>
                <value uuid="cce8f89a-a220-4e53-8734-5831a3a4eb77" user="admin"␣
↪timestamp="2010-09-08T15:36:01.836+02:00" change="VX-16299">2010-09-05T16:20:33Z</
↪value>
            </field>
            <field uuid="d9d9b21c-171d-402b-878d-cefa5b3f9727" user="admin" timestamp=
↪"2010-09-08T15:36:01.836+02:00" change="VX-16299">
                <name>sport_team</name>
                <value uuid="7493df98-be67-4fb6-97fe-a89b9e501207" user="admin"␣
↪timestamp="2010-09-08T15:36:01.836+02:00" change="VX-16299">Sweden</value>
            </field>
        </group>
    </timespan>
    <timespan start="-INF" end="+INF">
        <field uuid="7c5c49f9-c740-4b0a-93e8-81490fb65799" user="system" timestamp=
↪"2010-09-08T11:00:15.833+02:00" change="VX-16295">
            <name>shapeTag</name>
            <value uuid="9c2945d5-3480-436e-bfbb-2444e586961d" user="system"␣
↪timestamp="2010-09-08T11:00:15.833+02:00" change="VX-16295">original</value>
        </field>
    </timespan>
    <timespan start="2700" end="2940">
        <group uuid="0e9a54eb-0b90-4ed9-ac68-d2cb5d7abc73" user="admin" timestamp=
↪"2010-09-08T15:38:28.715+02:00" change="VX-16300">
            <name>goal</name>
            <field uuid="4e5ffd77-ab59-46fe-9939-47ab61df7523" user="admin" timestamp=
↪"2010-09-08T15:38:28.715+02:00" change="VX-16300">
                <name>sport_team</name>
                <value uuid="2a64f141-b3aa-4686-973c-7c254a0b77cb" user="admin"␣
↪timestamp="2010-09-08T15:38:28.715+02:00" change="VX-16300">Germany</value>
            </field>
            <field uuid="2444055e-40e0-49b6-8493-0f68df82f01a" user="admin" timestamp=
↪"2010-09-08T15:38:28.715+02:00" change="VX-16300">
                <name>sport_player</name>
                <value uuid="441404a4-882c-458a-af88-b2fad592d71c" user="admin"␣
↪timestamp="2010-09-08T15:38:28.715+02:00" change="VX-16300">Bob</value>
            </field>
            <field uuid="01d54bbb-d01e-4c6f-880e-1c1cbc4e598e" user="admin" timestamp=
↪"2010-09-08T15:38:28.715+02:00" change="VX-16300">
                <name>sport_time</name>
```

```
                <value uuid="71dd57e3-4a39-41ad-b351-78c4bc20ac0b" user="admin"␣
↪timestamp="2010-09-08T15:38:28.715+02:00" change="VX-16300">2010-09-05T16:42:05Z</
↪value>
            </field>
        </group>
    </timespan>
</MetadataDocument>
```

The metadata has now been corrected and contain the information that we wanted to record.

### 2.10.2 Defining a metadata schema

Based on the types in the *metadata example* we can specify a schema. There is no restriction in creating a Metadata Schema with different field-groups that contain same metadata-fields.

```
PUT /metadata-schema
Content-Type: application/xml

<MetadataSchemaDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <!-- The organization is optional and can exist [0,n] outside of groups -->
    <group name="organization" min="0" max="-1">
        <!-- An organization has one or more employees -->
        <group name="employee" min="1" max="-1" reference="false"/>
        <!-- An organization has zero or more projects -->
        <group name="project" min="0" max="-1" reference="false"/>
        <!-- An organization has exactly one name -->
        <field name="example_name" min="1" max="1" reference="false"/>
    </group>

    <!-- A project cannot exist outside of a group -->
    <group name="project" min="0" max="0">
        <!-- A project has at least one employee, which has to be referenced -->
        <group name="employee" min="1" max="-1" reference="true"/>
        <!-- A project has exactly one name -->
        <field name="example_name" min="1" max="1" reference="false"/>
        <!-- A project has exactly one location element (it still can have more than␣
↪one value) -->
        <field name="example_location" min="1" max="1" reference="false"/>
    </group>

    <!-- An employee cannot exist outside of a group -->
    <group name="employee" min="0" max="0">
        <!-- An employee has exactly one name -->
        <field name="example_name" min="1" max="1" reference="false"/>
        <!-- An employee might have a title -->
        <field name="example_title" min="0" max="1" reference="false"/>
    </group>
</MetadataSchemaDocument>
```

Retrieving the metadata of a new item:

```
GET /item/VX-11/metadata
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <item id="VX-11">
    <metadata>
      <revision>VX-47</revision>
```

```
      <timespan end="+INF" start="-INF">
        <field change="VX-47" timestamp="2010-12-17T13:15:04.495+01:00" user="system"␣
↪uuid="0f4ec1a0-8a5e-4b96-958a-b1ea7516e38a">
          <name>shapeTag</name>
          <value change="VX-47" timestamp="2010-12-17T13:15:04.495+01:00" user="system
↪" uuid="99a471ee-18fd-4440-a218-80a3df40b471">original</value>
        </field>
      </timespan>
    </metadata>
  </item>
</MetadataListDocument>
```

Validating it:

```
PUT /item/VX-11/metadata/validate
```

```
200 OK
```

Adding the organization in the example:

```
PUT /item/VX-11/metadata
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    <group>
      <name>organization</name>
      <field>
        <name>example_name</name>
        <value>My organization</value>
      </field>
      <group>
        <name>employee</name>
        <field>
          <name>example_name</name>
          <value>Bob</value>
        </field>
        <field>
          <name>example_title</name>
          <value>CEO</value>
        </field>
      </group>
      <group uuid="A">
        <name>employee</name>
        <field>
          <name>example_name</name>
          <value>Pete</value>
        </field>
        <field>
          <name>example_title</name>
          <value>Director</value>
        </field>
      </group>
      <group uuid="B">
        <name>employee</name>
        <field>
          <name>example_name</name>
          <value>Andrew</value>
```

```
        </field>
        <field>
          <name>example_title</name>
          <value>Editor</value>
        </field>
      </group>
      <group>
        <name>project</name>
        <field>
          <name>example_name</name>
          <value>Movie project</value>
        </field>
        <field>
          <name>example_location</name>
          <value>London</value>
          <value>Berlin</value>
        </field>
        <group>
          <name>employee</name>
          <reference>A</reference>
        </group>
        <group>
          <name>employee</name>
          <reference>B</reference>
        </group>
      </group>
    </group>
  </timespan>
</MetadataDocument>
```

```
200 OK
```

Adding an employee without a name to the organization:

```
PUT /item/VX-11/metadata
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    <group>
      <name>organization</name>
      <group mode="add">
        <name>employee</name>
        <field>
          <name>example_title</name>
          <value>Developer</value>
        </field>
      </group>
    </group>
  </timespan>
</MetadataDocument>
```

```
HTTP/1.1 400 An invalid parameter was entered
Context: metadata-schema
Reason: Too few of member example_name in group organization: 0 vs 1
```

### Alternate way of creating a schema

A schema can also be built when creating and modifying metadata field groups. To create the schema above, the following three requests can be made.

```
PUT /metadata-field/field-group/employee
Content-Type: application/xml

<MetadataFieldGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <schema min="0" max="0"/>
  <field>
    <name>example_name</name>
    <schema min="1" max="1" reference="false"/>
  </field>
  <field>
    <name>example_title</name>
    <schema min="0" max="1" reference="false"/>
  </field>
</MetadataFieldGroupDocument>
```

```
PUT /metadata-field/field-group/project
Content-Type: application/xml

<MetadataFieldGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <schema min="0" max="0"/>
  <field>
    <name>example_name</name>
    <schema min="1" max="1" reference="false"/>
  </field>
  <field>
    <name>example_location</name>
    <schema min="1" max="1" reference="false"/>
  </field>
  <group>
    <name>employee</name>
    <schema min="1" max="-1" reference="true"/>
  </group>
</MetadataFieldGroupDocument>
```

```
PUT /metadata-field/field-group/organization
Content-Type: application/xml

<MetadataFieldGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <schema min="0" max="-1"/>
  <field>
    <name>example_name</name>
    <schema min="1" max="1" reference="false"/>
  </field>
  <group>
    <name>employee</name>
    <schema min="1" max="-1" reference="false"/>
  </group>
  <group>
    <name>project</name>
    <schema min="0" max="-1" reference="false"/>
  </group>
</MetadataFieldGroupDocument>
```

CHAPTER

THREE

# COLLECTIONS AND LIBRARIES

This chapter describes collections and libraries, two concepts in Vidispine used to group items. The main differences between collections and libraries are:

- Collections can have metadata attached to them, libraries cannot.
- Collections can contain sub collections and can also contain libraries. Libraries can only contain items.
- Libraries can have dynamic content. You can attach an item search document to a library, and have it automatically update its content based on which items match the query.

Both can be assigned access controls and storage rules that apply to the items, and for collections, libraries and sub collections in them.

## 3.1 Collections

Collections are generic storage containers and can for example be used as:

- A sort of folder structure, where files are mapped as items and sub folders are mapped as sub collections in the hierarchy.
- A simple container for a number of items and collections.
- A representation of a Non Linear Editor (NLE) "bin".
- A representation of an entity in your domain.

### 3.1.1 Creating collections

Create a collection using *POST /collection*. Once created you can add items, libraries or other collections to it, add metadata or grant access to other users by adding access controls.

```
POST /collection?name=Pending%20review
```

```
<CollectionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <loc>http://localhost:8080/API/collection/VX-16</loc>
  <id>VX-16</id>
  <name>Pending review</name>
</CollectionDocument>
```

### 3.1.2 Searching for collections

You can search for collections in the same way as you can *search for items*.

```
PUT /collection
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <text>Pending</text>
</ItemSearchDocument>
```

```
<CollectionListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <hits>1</hits>
  <collection>
    <id>VX-16</id>
    <name>Pending review</name>
  </collection>
</CollectionListDocument>
```

### Searching for collections with specific items

Use an `item` query to find collections that contain specific items. For example, to find collections with a title containing 'Peach' or collections with items with similar titles:

```
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <operator operation="OR">
    <field>
      <name>title</name>
      <value>Peach</value>
    </field>
    <item>
      <field>
        <name>title</name>
        <value>Peach</value>
      </field>
    </item>
  </operator>
</ItemSearchDocument>
```

See *Joins on collection search*.

### Searching in a collection

You can also search for *items* in a collections using *PUT /collection/(collection-id)/item*. An alternative way of finding only items that exist in a collection is to query on the `__collection` transient metadata field. This is also more flexible as it allows you to find items in multiple collections, or using it as part of a more complex query.

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>__collection</name>
    <value>VX-16</value>
  </field>
  <operator operation="NOT">
    <field>
      <name>__collection</name>
      <value>VX-1</value>
    </field>
  </operator>
</ItemSearchDocument>
```

The difference between searching for items in a collection using *PUT /collection/(collection-id)/item* and *PUT /item* with a query on `__collection` is the default ordering, which is by collection order and by creation date, respectively.

There is also the `__ancestor_collection` transient metadata field that allows you to find items that exist in a collection or in a sub collection of that collection.

#### Listing collections that contain an item

If you want to see which collections contain an item, you can either look at the item metadata and look at the "`__collection`" field. There will be one entry for each collection that includes the item. This, however, does not take into account which collections a user has read access to. In order to see which collections contain an item with read permissions honored you can use `GET /item/(item-id)/collection`

### 3.1.3 Ordering collections

The entities in the collection are ordered, and new entities will be added at the end of the list. Use *POST /collection/(collection-id)/order* to change the order. The order will be enforced in requests to *GET /collection/(collection-id)* and *GET /collection/(collection-id)/item* for example.

To get the same ordering as in *GET /item* you will have to explicitly sort on the creation date, the `created` field, which is the default.

### 3.1.4 Update collection content

Items, libraries and collections can be added, removed and reordered in a collection in a single call. Use *GET /collection/(collection-id)* to get the content of the collection. Then rearrange, add and/or remove content and use *PUT /collection/(collection-id)* to update the collection.

### 3.1.5 Partial update collection content

If you only want to specify each change (add, move or remove entity) in a collection you can use *PUT /collection/(collection-id)* and specifying a mode in each content element.

```
PUT /collection/VX-2000
Content-Type: application/xml

<CollectionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <loc>http://localhost:8080/API/collection/VX-2000/</loc>
    <id>VX-2000</id>

    <!-- Set the name to "New name" -->
    <name>New name</name>

    <!-- Move item VX-3 after VX-2 -->
    <content mode="move" after="VX-2">
        <id>VX-3</id>
        <uri>http://localhost:8080/API/item/VX-3</uri>
        <type>item</type>
        <metadata/>
    </content>

    <!-- Add the items from library VX*40 to after item VX-2 -->
    <content mode="add" addItems="true" after="VX-2">
        <id>VX*40</id>
        <uri>http://localhost:8080/API/library/VX*40</uri>
        <type>library</type>
        <metadata/>
```

```
    </content>

    <!-- Add the library VX*44 before VX*33 -->
    <content mode="add" before="VX*33">
        <id>VX*44</id>
        <uri>http://localhost:8080/API/library/VX*44</uri>
        <type>library</type>
        <metadata/>
    </content>

    <!-- Remove the collection VX-500 -->
    <content mode="remove">
        <id>VX-500</id>
        <uri>http://localhost:8080/API/collection/VX-500</uri>
        <type>collection</type>
        <metadata/>
    </content>
</CollectionDocument>
```

### 3.1.6 Multiple relations between same entities

New in version 5.5.

In version 5.5, multiple relations between the same pair of collection and item, sub-collection or library are allowed. Each relation has a unique id (UUID), which can be used to reference relations instead of the entity id shown above.

In addition, a new mode is added: “update”, which only updates a relation, and not move it.

An example. Assume the following collection:

```
<CollectionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <id>VX-937</id>
    <name>coll</name>
    <content>
        <id>VX-718</id>
        <type>item</type>
        <reference>bbd5adbb-4826-49cc-8f26-b87a266f09db</reference>
        <metadata>
            <field>
                <key>myfield</key>
                <value>somedata</value>
            </field>
        </metadata>
    </content>
    <content>
        <id>VX-721</id>
        <type>item</type>
        <reference>9cf0a1ac-7c7b-49a7-acf9-33fce948e1aa</reference>
        <metadata/>
    </content>
    <content>
        <id>VX-718</id>
        <type>item</type>
        <reference>80eaad68-bcb7-4811-ada0-2fa1c366547c</reference>
        <metadata/>
    </content>
</CollectionDocument>
```

The following update:

- Move the last VX-718 to the top.
- Removes the first VX-718.
- Adds two new VX-719 (at the end of the list).
- Sets metadata on VX-721.

```
<CollectionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <content mode="move" before="bbd5adbb-4826-49cc-8f26-b87a266f09db"/>
        <reference>80eaad68-bcb7-4811-ada0-2fa1c366547c</reference>
    </content>
    <content mode="remove">
        <reference>bbd5adbb-4826-49cc-8f26-b87a266f09db</reference>
    </content>
    <content mode="add">
        <id>VX-719</id>
        <type>item</type>
    </content>
    <content mode="add">
        <id>VX-719</id>
        <type>item</type>
    </content>
    <content mode="update">
        <reference>9cf0a1ac-7c7b-49a7-acf9-33fce948e1aa</reference>
        <metadata>
            <field>
                <key>myfield</key>
                <value>newdata</value>
            </field>
        </metadata>
    </content>
</CollectionDocument>
```

### 3.1.7 Metadata on collection to entity relations

Each collection to entity relation can contain metadata which is stored in fields with a key and a value and can be modified when updating the collection.

```
GET /collection/VX-3866
```

```
<CollectionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <loc>http://localhost:8080/API/collection/VX-3866/</loc>
  <id>VX-3866</id>
  <content>
    <id>VX-5480</id>
    <uri>http://localhost:8080/API/item/VX-5480</uri>
    <type>item</type>
    <metadata/>
  </content>
  <content>
    <id>VX*1810</id>
    <uri>http://localhost:8080/API/library/VX*1810</uri>
    <type>library</type>
    <metadata/>
  </content>
</CollectionDocument>
```

```
PUT /collection/VX-3866
Content-Type: application/xml

<CollectionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <loc>http://localhost:8080/API/collection/VX-3866/</loc>
  <id>VX-3866</id>
  <content>
    <id>VX-5480</id>
    <uri>http://localhost:8080/API/item/VX-5480</uri>
    <type>item</type>
    <metadata>
      <field>
        <key>AddedBy</key>
        <value>Jane Doe</value>
      </field>
    </metadata>
  </content>
  <content>
    <id>VX*1810</id>
    <uri>http://localhost:8080/API/library/VX*1810</uri>
    <type>library</type>
    <metadata>
      <field>
        <key>AddedBy</key>
        <value>John Doe</value>
      </field>
    </metadata>
  </content>
</CollectionDocument>
```

### 3.1.8 Collections as folders

As mentioned in the introduction, collections could be used to represent folders, as a way for your users to organize their items.

This could be an entirely logical grouping, or correspond to the actual directory structure of the items files on the file system. To achieve the later, you can mark the collections as folder mapped collections. See *Folder mapped collections* in the API reference on how to set this up.

### 3.1.9 Representative thumbnails

A new metadata field has been added to the collection metadata: `representativeItems`. This field can contain a list of items that will represent this collection. Vidispine will automatically get the representative thumbnails of each item and add a transient metadata field on the collection metadata.

For example:

```
<field>
  <name>representativeItems</name>
  <value>VX-653</value>
  <value>VX-657</value>
  <value>VX-658</value>
  <value>VX-659</value>
</field>
```

will add those values to the metadata:

```
<field>
    <name>__representativeThumbnails</name>
    <value>/API/thumbnail/VX-2/VX-653;version=0/2100@NTSC30</value>
    <value>/API/thumbnail/VX-2/VX-657;version=0/0@24000</value>
    <value>/API/thumbnail/VX-2/VX-658;version=0/3300297@30000</value>
    <value>/API/thumbnail/VX-2/VX-659;version=0/500@PAL</value>
</field>
<field>
    <name>__representativeThumbnailsNoAuth</name>
    <value>/APInoauth/thumbnail/VX-2/VX-653;version=0/2100@NTSC30?
↪hash=9dfb29f8159532b1d3a119462e64c03f</value>
    <value>/APInoauth/thumbnail/VX-2/VX-657;version=0/0@24000?
↪hash=bb75e99dd2f1f961810c85fab99cd75f</value>
    <value>/APInoauth/thumbnail/VX-2/VX-658;version=0/3300297@30000?
↪hash=696fa412368ccc0ac11fc30018ea8062</value>
    <value>/APInoauth/thumbnail/VX-2/VX-659;version=0/500@PAL?
↪hash=0737888e52cb4e66041ba7d1e58b22be</value>
</field>
```

In combination with *Stitching images*, this can be used to easily create and cache a collection thumbnail without having to track the item update notifications.

## 3.2 Libraries

Whereas collections are more of a generic container for entities, the strength of libraries lies in the ability to have the library content dynamically updated based on a query.

Use libraries to for example:

- Manage the current search performed by a user.
- To represent saved searches created by your users.
- To implement dynamic storage rules or access control restrictions based on the metadata of items.

### 3.2.1 Creating libraries

When searching for items you can create a library containing the items matching the query by specifying `result=library`. If used together with `autoRefresh=true` you can create a "saved search". When accessing the library later, its content will return

```
PUT /item?updateMode=REPLACE&autoRefresh=TRUE&result=library HTTP/1.1
Accept: application/xml
Content-type: application/xml
```

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>project_priority</name>
    <value>urgent</value>
  </field>
</ItemSearchDocument>
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <library>VX*19</library>
  <item>VX-13</item>
  <item>VX-14</item>
```

```
  <item>VX-71</item>
</ItemListDocument>
```

Check the library settings to find out how a library was created, or why a library contains a specific set of items, for example when using self-refreshing libraries.

```
GET /library/VX*67/settings
```

```
<LibrarySettingsDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <id>VX*67</id>
   <username>admin</username>
   <updateMode>REPLACE</updateMode>
   <autoRefresh>true</autoRefresh>
   <query>
      <field>
      <name>originalWidth</name>
      <range>
         <value>640</value>
         <value>720</value>
      </range>
      </field>
   </query>
</LibrarySettingsDocument>
```

#### Libraries without a query

Libraries can also be created using *POST /library*. You will need to specify the items that the library should contain, but this can also be changed afterwards.

```
POST /library HTTP/1.1
Content-Type: application/xml

<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item id="VX-250"/>
   <item id="VX-1000"/>
</ItemListDocument>
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <uri>VX*48</uri>
</URIListDocument>
```

Check the library settings and you will see that it does not specify a query, in comparison to libraries created when searching.

### 3.2.2 Automatic deletion

Libraries will be automatically deleted after having not being accessed for a period of 24 hours. There are some exception to this rule. If any of the following conditions apply, the library will not be automatically deleted:

- The library is part of a collection
- The library has a storage rule set
- The library has a site rule set
- The library has `autoRefresh=true`
- The library has an `updateFrequencey` set.

### 3.2.3 Self-refreshing libraries

Libraries can be set to keep their contents up to date with the queries in two ways (see the table below). The two different methods can either be used together or separately. Neither of these modes will have an affect on transient libraries, as they will always be kept up to date.

| Name | Values | Description |
|---|---|---|
| `autoRefresh` | `true` or `false` (default) | If `true`, items will be tracked as their metadata is modified. |
| `updateFrequency` | positive integer | If set, the library will be rebuilt periodically. The integer describes the minimum time, in minutes, between updates. |

Having `autoRefresh` set means that metadata changes will have an almost immediate effect on libraries. But it has the drawback that libraries using variables, such as a timestamp search containing ranges with the "NOW" variable, will not be updated unless a user changes its metadata. To remedy this libraries can be updated periodically. From a performance point of view though, it is more efficient to check if an item belongs to a library then to refresh an entire library – so period updates should be done with care.

> **Caution:** *Queries involving variables*
>
> Using variables in queries, e.g. the use of the word "NOW" when searching timestamped metadata, is not reliable for libraries unless they are either set as TRANSIENT or they are set to be updated periodically.

#### Update modes

**MERGE** In this mode any items that matches query will be added to the library without removing any existing items.

**REPLACE** Unlike MERGE, this mode will also remove items that no longer matches the query.

**TRANSIENT** This mode has the same semantics as REPLACE, with some important differences. It only contains items on a logical basis, so instead of simply retrieving its items it needs to perform a search every time its contents is being requested. This leads to a faster creation time than REPLACE, but slower lookup and cannot be used to *restrict item access*.

#### Example

Creating a library that contains items created within the last 5 days.

```
PUT /item?autoRefresh=false&updateFrequency=60&updateMode=REPLACE&result=library
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <field>
      <name>created</name>
      <range>
         <value>NOW-5DAYS</value>
         <value>NOW</value>
      </range>
   </field>
</ItemSearchDocument>
```

#### Restrictions

At most 999 self-refreshing libraries can exist in the system simultaneously.

If using the default Solr configuration, it is a good idea to set the *useLucene* property to speed up matching of self-refreshing libraries.

### 3.2.4 Restricting access to items

Setting access controls on a library will cascade down on the items. This means that libraries can be used to batch update access controls on a set of items. Note that this does not work on libraries with `updateMode TRANSIENT`.

### 3.2.5 Storage rules on libraries

You can set storage rules on libraries. All items belonging to the library will then be affected by the rule. Note that this does not work on libraries with `updateMode TRANSIENT`. Having a storage rule on a library will also prevent it from being automatically deleted.

CHAPTER

FOUR

# SHAPES, COMPONENTS AND TRANSCODING

## 4.1 Item shapes

A shape is a physical representation of an item. Each shape is made up out of one or more components that correspond to content of a file.

### 4.1.1 Shapes

Each item will typically have at least a single shape, the *original* shape, along with one or more alternate representations of the asset.

- For video, this can be a low-resolution version, a web version and a mobile version. Another example is if you have multiple versions of the same video, but each with different audio or text tracks. Those versions would then be separate shapes.
- For text this could be the word processor document format, a PDF or a plain text version.

You will find that the information extracted and presented for video and audio files is richer than what's provided for other type of files, such as PDFs or zip files. For the former information about the container and video- and audio streams is provided, while the latter is typically presented as a shape with a binary component.

To distinguish between different shapes you can use tags. These are described in the *Shape tags and presets* section.

#### Importing shapes

Vidispine will create an `original` shape when an item is first *imported*. To import additional shapes to an item, for example files created by an external transcoder, then that can be done by creating a `SHAPE_IMPORT` job.

A shape import job will:

- Transfer content to a Vidispine supervised storage.
- Media check the imported file.
- Create a new shape and add it to the item.

Use the *item shape resource* to import new shapes. An image could for example be imported to and item, and tagged with the `large-jpg` tag, using:

```
POST /item/VX-12/shape?uri=file:///srv/ftp/the-doctor.jpg&tag=large-jpg
```

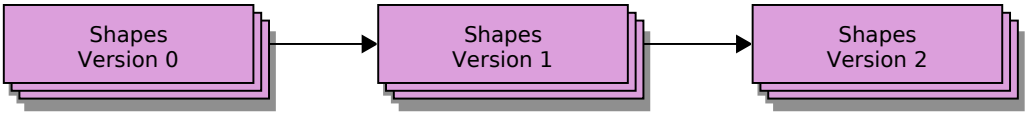
```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-169826</jobId>
  <user>admin</user>
  <started>2014-07-03T18:21:09.795Z</started>
  <status>READY</status>
  <type>SHAPE_IMPORT</type>
```

```
  <priority>MEDIUM</priority>
</JobDocument>
```

### 4.1.2 Essence versions

If you have assets that change over time, and wish to track all of those versions, then you can use an item to represent the asset and then import each update to the asset as a new essence version on the item.

Vidispine will return the shapes and thumbnails for the latest essence version by default, but you can of course select to have older versions returned as well.




#### Creating a new essence version

New essence versions are created using `ESSENCE_VERSION` jobs. See *Essence versions* for the different ways of starting such jobs. For example, creating a new essence version for an item by providing the location of the new asset.

```
POST /item/VX-37/shape/essence?uri=file:///home/lisa/render-1.jpg
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-39</jobId>
  <user>lisa</user>
  <started>2014-07-03T06:52:45.114Z</started>
  <status>READY</status>
  <type>ESSENCE_VERSION</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

This new image will then show up as the original shape of the item, and will be used as the input on any future transcodes. By viewing the shape we can see that this shape belongs to a new essence version. Note that the essence version numbers are zero-based.

```
<ItemDocument xmlns="http://xml.vidispine.com/schema/vidispine" id="VX-37">
  <shape>
    <id>VX-38</id>
    <essenceVersion>1</essenceVersion>
    <tag>original</tag>
    <mimeType>image/jpeg</mimeType>
    <containerComponent>...</containerComponent>
    <videoComponent>...</videoComponent>
    <metadata>
      <field>
        <key>originalFilename</key>
        <value>render-1.jpg</value>
      </field>
    </metadata>
  </shape>
</ItemDocument>
```

We can also see that there's a new essence version by retrieving the essence versions for the items.

```
GET /item/VX-37/shape/version
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>http://localhost:8080/API/item/VX-37/shape/version/1</uri>
  <uri>http://localhost:8080/API/item/VX-37/shape/version/0</uri>
</URIListDocument>
```

### 4.1.3 Transcoding

An item can be transcoded either when it is *imported* or afterwards by using the item transcode resource. When transcoding an already imported item a `TRANSCODE` job will be used. A transcode job will:

- Create any new entities, such as the new files that are about to appear.
- Create a transcoding task and submits it to a transcoder.
- Media check the new files and update the item.

The difference between transcoding while and after importing is that the former can be done in parallel to any transfers that may be needed, while the latter is a serial task as the input files, the files from the original shape of the item, should already exist on a storage managed by Vidispine.

#### Starting transcode jobs

The transcodes to perform are identified using *shape tags* that contains the *transcode preset* that the defines the desired outputs.

Use the `tag` parameter when starting an import job to transcode an item while it is being imported. See *Transcoding* for more information on the subject.

```
POST /import?uri=file:///srv/incoming/media.mov&tag=lowres,android
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-169819</jobId>
  <user>admin</user>
  <started>2014-07-03T07:20:14.220Z</started>
  <status>READY</status>
  <type>PLACEHOLDER_IMPORT</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

To transcode an existing item, use the *transcode resource* with the `tag` parameter as above.

```
POST /item/VX-191440/transcode?tag=lowres,android
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-169820</jobId>
  <user>admin</user>
  <started>2014-07-03T07:22:47.900Z</started>
  <status>READY</status>
  <type>TRANSCODE</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

### Transcode progress

The progress of the transcode is available from the job, both as progress on the transcode step and as key-value metadata on the job.

```
<task id="237">
  <step>200</step>
  <attempts>0</attempts>
  <status>STARTED_ASYNCHRONOUS</status>
  <timestamp>2014-07-03T07:27:58.030Z</timestamp>
  <description>Transcoding.</description>
  <progress total="100" unit="percent">75.0</progress>
  <subStep>
    <timestamp>2014-07-03T07:22:48.051Z</timestamp>
    <description>Starting transcode</description>
  </subStep>
</task>
```

And from the job metadata, where you will find the `transcode*` job metadata, that also includes the estimated time left and the progress expressed in the media time.

```
<data>
  <key>transcodeDurations</key>
  <value>8000000@1000000</value>
</data>
<data>
  <key>transcodeMediaTimes</key>
  <value>288000@48000</value>
</data>
<data>
  <key>transcodeProgress</key>
  <value>75.0</value>
</data>
<data>
  <key>transcodeEstimatedTimeLeft</key>
  <value>6.2072</value>
</data>
<data>
  <key>transcodeWallTime</key>
  <value>18.6216</value>
</data>
```

## 4.1.4 Thumbnailing

Thumbnails are by default created if an item is transcoded while being imported. To create thumbnails or posters for an item, use a `THUMBNAIL` job. A thumbnail job will:

- Create a thumbnailing task and submit it to a transcoder.
- Update the representative thumbnail of the item.

The location of the representative thumbnail is stored in the item metadata, so if you wish to present a number of items to a user, along with a thumbnail of each item, then it is recommended that you read the thumbnails from the `representativeThumbnail` metadata field instead of fetching all thumbnails for all items. There is also the `representativeThumbnailNoAuth` field that provides the thumbnail at a location that does not require authentication.

```
<timespan start="-INF" end="+INF">
  <field uuid="b578cfe7-cf8b-476f-866f-7027e0dba542" user="system" timestamp="2014-07-
↪03T09:23:24.172+02:00" change="VX-1345770">
```

```
    <name>representativeThumbnail</name>
    <value uuid="a159d13c-4a70-4e6a-83fd-36a7b0ef25af" user="system" timestamp="2014-
↪07-03T09:23:24.172+02:00" change="VX-1345770">/API/thumbnail/VX-2/VX-191440;
↪version=0/0@PAL</value>
  </field>
  <field uuid="12c6553d-7473-42bf-95b4-875afd1cac74" user="system" timestamp="2014-07-
↪03T10:46:40.728+02:00" change="VX-1345771">
    <name>representativeThumbnailNoAuth</name>
    <value uuid="9b70a04c-8755-426b-9446-3f4857cb87e1" user="system" timestamp="2014-
↪07-03T10:46:40.728+02:00" change="VX-1345771">/APInoauth/thumbnail/VX-2/VX-191440;
↪version=0/0@PAL?hash=b5424e9878940a333b6230817ae88eef</value>
  </field>
</timespan>
```

### Starting a thumbnail job

Use the *thumbnail resource* to create thumbnail jobs for an item.

```
POST /item/VX-191440/thumbnail?createThumbnails=true
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-169821</jobId>
  <user>admin</user>
  <started>2014-07-03T08:46:09.960Z</started>
  <status>READY</status>
  <type>THUMBNAIL</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

The thumbnails will be uploaded to the item as the job progresses. You can find them by inspecting the item. For example:

```
GET /item/VX-191440?content=thumbnail
```

```
<ItemDocument xmlns="http://xml.vidispine.com/schema/vidispine" id="VX-191440">
  <thumbnails>
    <uri>http://localhost:8080/API/thumbnail/VX-2/VX-191440;version=0/0@PAL</uri>
  </thumbnails>
</ItemDocument>
```

If you wish to see which thumbnails were created by a specific thumbnail job, then you can check the `thumbnails` job metadata.

```
<data>
  <key>thumbnails</key>
  <value>{"[TC:0@PAL]":"http://localhost:8080/API/thumbnail/VX-2/VX-191440;version=0/
↪0@25"}</value>
</data>
```

## 4.1.5 Analyzing media

Shapes can be analyzed to detect for example detect cropping and silence. See *Shape analysis*.

## 4.2 Shape tags and presets

Shapes can be tagged in order to retrieve their file contents easily using *Retrieving item information*. The system adds certain tags to shapes automatically during certain operations, such as an import job. Predefined tags can be seen in the table below.

| Tag | Description |
|---|---|
| original | The first shape that was created for the item. |

While shape tags serve as a "name tag" for shapes, they also contain the recipe for how new shape instances with the shape tag should be constructed, or transcoded, from other shapes. This is defined using a *transcode preset* that defines the format, codec, bitrate etc of the shapes.

### 4.2.1 Transcode presets

The transcode preset specifies the output format, codec and encoding settings that should be used when transcoding. You can either

- Use one of the built in *preset templates*.
- Use one of the presets *defined in this documentation*.
- Define your own preset. See *Transcode preset elements* for more information.

#### Preset templates

Vidispine comes with some preset templates built in. These can be added to the system by making a `PUT` request to `APIinit/preset-templates`. These template tags have names that begin with double underscore and cannot be overwritten (Also, shape tags with names starting with double underscore cannot be added to the system).

### 4.2.2 Scripting transcode presets

Transcode presets can be made dynamic by assigning a JavaScript to them. Made available to the script will be the shape that is going to be transcoded as well as the unmodified preset. The shape can be used as input to determine for example the original resolution of the media. For output the preset can be modified before it is sent to the transcoder. An overview is given in the table below.

| Mode | Identifier | XML Type | Java Type |
|---|---|---|---|
| input | `jobMetadata` | - | `java.util.Map<String,String>` |
| input | `metadata` | *MetadataType* | `com.vidispine.generated.MetadataType` |
| input | `shape` | *ShapeType* | `com.vidispine.generated.ShapeType` |
| output | `preset` | *TranscodePresetType* | `com.vidispine.generated.TranscodePresetType` |

The given data types are generated from the XML schema and belong to the package `com.vidispine.generated`. They follow JavaBean standard, i.e. getters and setters for their attributes.

**Caution:** *Lists of integer*

When adding integers of a list, simply using integer literals will not work. Instead `java.lang.Integer` must be used, for example: `list.add(new java.lang.Integer(5));`

**Tip:** When writing the script, or after you have written one, write tests that verifies it for a range of shapes using the *script test resource*.

### Examples

#### A preset that only produces two audio channels in the output

First we create a preset with only the formats and codecs set.

```
PUT /shape-tag/h264
Content-Type: application/xml

<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mp4</format>
  <audio>
    <codec>aac</codec>
  </audio>
  <video>
    <codec>h264</codec>
  </video>
</TranscodePresetDocument>
```

```
200 OK
```

Then we add the script

```
PUT /shape-tag/h264/script
Content-Type: application/javascript

// Retrieve the channel count: <ShapeDocument><audioComponent><channelCount>
var channelCount = shape.getAudioComponent().get(0).getChannelCount();

// If we have more than two channels, limit it to the first two:
if (channelCount > 2) {
    // Adding elements to <TranscodePresetDocument><audio><channel>
    preset.getAudio().getChannel().add(new java.lang.Integer(0));
    preset.getAudio().getChannel().add(new java.lang.Integer(1));
}
```

```
200 OK
```

The result preset will then look like this if the input shape has more than two audio channels:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mp4</format>
  <audio>
    <codec>aac</codec>
    <channel>0</channel>
    <channel>1</channel>
  </audio>
  <video>
    <codec>h264</codec>
  </video>
</TranscodePresetDocument>
```

#### Scaling the output depending on the input

Using the same shape-tag as in the example above we can use the following script.

```
// Retrieve the width and height of the input
var width = shape.getVideoComponent().get(0).getResolution().getWidth();
var height = shape.getVideoComponent().get(0).getResolution().getHeight();

if (width == 720 && height == 608) {
    // Create the scaling element
    var scaling = new com.vidispine.generated.ScalingType();
    preset.getVideo().setScaling(scaling);

    // Crop 32 pixels from the top
    scaling.setTop(32);

    // Set the desired display aspect ratio
    var targetDar = new com.vidispine.generated.AspectRatioType();
    targetDar.setHorizontal(4);
    targetDar.setVertical(3);
    scaling.setTargetDAR(targetDar);

    // Set the desired resolution
    scaling.setWidth(480);
    scaling.setHeight(360);
} else if (height > 700) {
    // Create the scaling element
    var scaling = new com.vidispine.generated.ScalingType();
    preset.getVideo().setScaling(scaling);

    // Set the desired display aspect ratio
    var targetDar = new com.vidispine.generated.AspectRatioType();
    targetDar.setHorizontal(16);
    targetDar.setVertical(9);
    scaling.setTargetDAR(targetDar);

    // Set the desired resolution
    scaling.setWidth(640);
    scaling.setHeight(360);
} else {
    // Create the scaling element
    var scaling = new com.vidispine.generated.ScalingType();
    preset.getVideo().setScaling(scaling);

    // Set the desired display aspect ratio
    var targetDar = new com.vidispine.generated.AspectRatioType();
    targetDar.setHorizontal(4);
    targetDar.setVertical(3);
    scaling.setTargetDAR(targetDar);

    // Set the desired resolution
    scaling.setWidth(320);
    scaling.setHeight(240);
}
```

### 4.2.3 Transcode preset elements

This section highlights some of the settings and possibilities that are often useful when authoring a transcode preset.

### Setting a preferred source tag

It is possible to specify that another file than the original should be used as the source file when transcoding. This is done using the `preferredSourceTag` element.

### Burning in the timecode in the video

Vidispine can burn the timecode into the output video. To enable this, set the `burnTimecode` element to true within the `video` element.

### Customizing the timecode

The following settings can be added as key-value pairs in the transcode preset:

- `bitc_font`. Font to use.

  **`monospace`** fixed-width font (default)

  **`sans`** sans-serif font

  **`serif`** font with serifs

- `bitc_size`. Height of font in pixels. Defaults to 15th of the video height.
- `bitc_sizeRel`. Height of font relative to full video frame.
- `bitc_xRel`. Position relative to width of video. Default is `0.5` (middle).
- `bitc_yRel`. Position relative to height of video. Default is `0.9` (bottom).
- `bitc_horizontalbase`. Horizontal position of base point relative to text bounding box.

  **0.0 (or `left`)** base point is left border of bounding box.

  **0.5 (or `center`)** base point is center of bounding box.

  **1.0 (or `right`)** base point is right border of of bounding box.

- `bitc_verticalbase`. Vertical position of base point relative to text bounding box.

  **0.0 (or `top`)** base point is top border of of bounding box.

  **0.5 (or `middle`)** base point is middle of bounding box.

  **1.0 (or `bottom`)** base point is bottom border of bounding box.

- `bitc_r`, `bitc_g`, `bitc_b`, `bitc_a`. RGB+alpha of text. Defaults to opaque white.
- `bitc_outlineR`, `bitc_outlineG`, `bitc_outlineB`, `bitc_outlineR`. RGB+alpha of surrounding block. Defaults to opaque black.
- `bitc_outline`. Type of outline. Default is `bar`.
- `bitc_outlinesize`. Size (margin) of outline.

### Setting a maximum duration of a chunk in QuickTime files

It is possible to specify a maximum duration for chunks in QuickTime files (MOV/MP4/3GPP). To set the duration add the `maxChunkDuration` element to the `TranscodePresetType`.

### Example: setting the maximum chunk duration to 2 seconds

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mp4</format>
  <maxChunkDuration>
    <samples>50</samples>
    <timeBase>
        <numerator>1</numerator>
        <denominator>25</denominator>
    </timeBase>
  </maxChunkDuration>
  <audio>
    <codec>aac</codec>
    <bitrate>320000</bitrate>
  </audio>
  <video>
    <codec>h264</codec>
    <bitrate>500000</bitrate>
    <framerate>
      <numerator>1</numerator>
      <denominator>25</denominator>
    </framerate>
    <resolution>
      <width>512</width>
      <height>288</height>
    </resolution>
  </video>
</TranscodePresetDocument>
```

### Mixing audio channels

It is possible to define advanced mappings between input and output audio channels. This is done using the `mix` element.

### Example: mixing 5.1 audio into stereo

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mp4</format>
  <audio>
    <codec>aac</codec>
    <bitrate>128000</bitrate>
    <mix>
        <input channel="0" stream="1" gain="1.0"/>
        <input channel="2" stream="1" gain="0.5"/>
        <input channel="4" stream="1" gain="1.0"/>
    </mix>
    <mix>
        <input channel="1" stream="1" gain="1.0"/>
        <input channel="2" stream="1" gain="0.5"/>
        <input channel="5" stream="1" gain="1.0"/>
    </mix>
  </audio>
  <video>
    <scaling>
      <width>512</width>
      <height>288</height>
    </scaling>
    <codec>h264</codec>
```

```
    <bitrate>256000</bitrate>
    <framerate>
      <numerator>1</numerator>
      <denominator>25</denominator>
    </framerate>
  </video>
</TranscodePresetDocument>
```

The value of the `stream` attribute can be deduced from the input shape. The `gain` attribute is expressed linearly, i.e. a value of 1.0 means a gain of 0 dB. Also, since the number of input channels will probably vary with different inputs, this functionality is best utilized in conjunction with the JavaScript functionality described below.

The `mix` element can also be used to insert silent audio channels in the output.

#### Example: adding two silent audio channels in output

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mp4</format>
  <audio>
    <codec>aac</codec>
    <bitrate>128000</bitrate>
    <mix silence="true"/>
    <mix silence="true"/>
  </audio>
  <video>
    <scaling>
      <width>512</width>
      <height>288</height>
    </scaling>
    <codec>h264</codec>
    <bitrate>256000</bitrate>
    <framerate>
      <numerator>1</numerator>
      <denominator>25</denominator>
    </framerate>
  </video>
</TranscodePresetDocument>
```

### Splitting audio channels to mono files

It is possible to split audio channels into separate mono audio files. And they can be *renamed according to their channel ids* .

#### Example: split specific channels

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>wav</format>
  <audio>
    <codec>pcm_s16le</codec>
    <channel>0</channel>
    <channel>3</channel>
    <channel>5</channel>
    <monoFile>true</monoFile>
  </audio>
  <video>
    <noVideo>true</noVideo>
```

```
  </video>
</TranscodePresetDocument>
```

### Example: split all channels

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>wav</format>
  <audio>
    <codec>pcm_s16le</codec>
    <monoFile>true</monoFile>
    <allChannel>true</allChannel>
  </audio>
  <video>
    <noVideo>true</noVideo>
  </video>
</TranscodePresetDocument>
```

### Splitting audio channels to different output files

It is possible to split audio channels into files that contain more than one channels. And they can be *renamed according to their channel ids*.

### Example

This preset below will produce three files:

1. A WAV file containing 1 audio stream with 2 channels.
2. A MOV file containing 2 audio streams, each stream has one channel.
3. A MP4 file containing only the video.

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mp4</format>
  <audio>
    <output>
      <format>wav</format>
      <codec>pcm_s24le</codec>
      <channel>0</channel>
      <channel>1</channel>
    </output>
    <output>
      <format>mov</format>
      <codec>aac</codec>
      <bitrate>320000</bitrate>
      <channel>2</channel>
      <channel>3</channel>
      <stream>1</stream>
      <stream>1</stream>
    </output>
  </audio>
  <video>
    <codec>h264</codec>
    <bitrate>1000000</bitrate>
    <framerate>
      <numerator>1</numerator>
      <denominator>25</denominator>
```

```
    </framerate>
    <resolution>
      <width>512</width>
      <height>288</height>
    </resolution>
  </video>
</TranscodePresetDocument>
```

### Specifying multiple audio tracks with different audio codecs in output

It is possible to output multiple audio tracks, with different audio codecs, using the `audioTrack` element in the preset. The `AudioTrackTranscodePresetType` has the `channel` and `mix` elements from the `AudioTranscodePresetType`. However, there is no `stream` element, since adding multiple `AudioTrackTranscodePresetType` gives the same behaviour.

If you add a *script* to your preset you can access the `audioTrack` list with: `preset.getAudioTrack()`. This is useful for doing advanced channel mappings and mixdown depending on the input(s).

### Example: two audio tracks, one with AC-3 and one with AAC

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <format>mov</format>
    <audioTrack>
        <codec>ac3</codec>
        <bitrate>384000</bitrate>
    </audioTrack>
    <audioTrack>
        <codec>aac</codec>
        <bitrate>96000</bitrate>
    </audioTrack>
</TranscodePresetDocument>
```

### Example: two audio tracks, one with AC-3 and one with a mixdown to stereo AAC

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <format>mov</format>
    <audioTrack>
        <codec>ac3</codec>
        <bitrate>384000</bitrate>
    </audioTrack>
    <audioTrack>
        <codec>aac</codec>
        <bitrate>96000</bitrate>
        <mix>
            <input channel="0" gain="1.0"/>
            <input channel="2" gain="0.5"/>
            <input channel="4" gain="1.0"/>
        </mix>
        <mix>
            <input channel="1" gain="1.0"/>
            <input channel="2" gain="0.5"/>
            <input channel="5" gain="1.0"/>
        </mix>
```

```
    </audioTrack>
</TranscodePresetDocument>
```

### Example: two audio tracks, one with AC-3 and one with AAC, picking some channels

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <format>mov</format>
    <audioTrack>
        <codec>ac3</codec>
        <bitrate>384000</bitrate>
        <channel>0</channel>
        <channel>1</channel>
    </audioTrack>
    <audioTrack>
        <codec>aac</codec>
        <bitrate>96000</bitrate>
        <channel>2</channel>
        <channel>3</channel>
    </audioTrack>
</TranscodePresetDocument>
```

### Specifying audio channel layout

New in version 5.6.

It is possible to specify the channel layout by adding a `<channelLayoutName>` or `<channelLayout>`

**Note:** Currently not supported by the nablet_aac codec

`<channelLayout>` is a binary representation of the channel layout.

`<channelLayoutName>` accepts an layout name.

The name can be:

- an usual channel layout (mono, stereo, 4.0, quad, 5.0, 5.0(side), 5.1, 5.1(side), 7.1, 7.1(wide), downmix)
- the name of a single channel (FL, FR, FC, LFE, BL, BR, FLC, FRC, BC, SL, SR, TC, TFL, TFC, TFR, TBL, TBC, TBR, DL, DR)
- a multiple of the above separated with '+' or 'l', for example: "stereo+FC" or "FL+FR+FC"

### Example: 5.1

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    ...
    <audio>
      ...
      <channelLayout>63</channelLayout>
    </audio>
</TranscodePresetDocument>
```

or

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    ...
    <audio>
      ...
      <channelLayoutName>"5.1"</channelLayoutName>
    </audio>
</TranscodePresetDocument>
```

### Image overlays

One can overlay images on the output at specific positions and intervals by using the `overlay` element. Note that the image is overlaid as is and will not be scaled in any way, meaning that you may want to overlay different images depending on the output resolution. Specifying an overlay interval in an image preset is not supported. Only PNG overlays are supported.

Multiple overlays are supported.

### Example: overlaying a logo

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  ...
  <overlay>
    <uri>http://example.com/logo.png</uri>
    <x>10</x>
    <y>30</y>
  </overlay>
</TranscodePresetDocument>
```

### Text overlays

In addition to image overlays, test overlays can also be added. The XML syntax is somewhat different from the image overlay syntax, see TextOverlayType in *XML Schema* for details.

### Example: overlaying text

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  ...
  <textOverlay>
    <text xRel="0.1" yRel="0.9" horizontalBase="left" align="left" verticalBase=
↪"bottom" sizeRel="0.05">
      <line>Happy</line>
      <line>birthday!</line>
    </text>
  </textOverlay>
  <textOverlay>
    <text xRel="0.9" yRel="0.9" horizontalBase="right" align="right"
          verticalBase="bottom" sizeRel="0.02" r="0" g="0" b="0" language="ar">
      <line>&#1571;&#1580;&#1605;&#1604; &#1575;&#1604;&#1571;&#1605;&#1606;&#1610;&
↪#1575;&#1578;&#1616; &#1576;&#1575;&#1604;&#1605;&#1586;&#1610;&#1583; &#1605;&
↪#1606; &#1575;&#1604;&#1587;&#1593;&#1575;&#1583;&#1577; &#1608;&#1575;&#1604;&
↪#1607;&#1606;&#1575;&#1569;!</line>
    </text>
  </textOverlay>
</TranscodePresetDocument>
```

### Thumbnails at specific interval

By adding the element `<thumbnailPeriod>` to the preset, the interval of the the thumbnails can be customized. The default interval is 10 seconds. Example, for one thumbnail per second:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    ...
    <thumbnailPeriod>
        <samples>1</samples>
        <timeBase>
            <numerator>1</numerator>
            <denominator>1</denominator>
        </timeBase>
    </thumbnailPeriod>
    ...
</TranscodePresetDocument>
```

### Thumbnails at scene changes

By adding the element `<thumbnailPlugin>scenechange</thumbnailPlugin>` to the preset, the fixed thumbnail interval is replaced by an algorithm that extracts thumbnail at scene changes in the version.

Since 4.14, you can combine this element with `<thumbnailPeriod>` to specify the maximum interval between thumbnails. The example below will create a thumbnail at every scene change, but never more than 20 seconds apart:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    ...
    <thumbnailPeriod>
        <samples>1</samples>
        <timeBase>
            <numerator>1</numerator>
            <denominator>1</denominator>
        </timeBase>
    </thumbnailPeriod>
    <thumbnailPlugin>scenechange</thumbnailPlugin>
    ...
</TranscodePresetDocument>
```

### MPEG-DASH representation presets

New in version 5.4.

When creating presets for MPEG-DASH representations, the format should be set as `mpd-representation`.

Video representations should have `noAudio` and audio representations should have `noVideo`.

### Example

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <name>__dash_1080p</name>
    <format>mpd-representation</format>
    <audio>
        <noAudio>true</noAudio>
    </audio>
    <video>
        ...
    </video>
```

```
    <metadata/>
</TranscodePresetDocument>
```

### Deinterlacing video

By adding the setting `deinterlacer` with value `advanced` the video will be deinterlaced.

The deinterlacer takes some other settings:

- `deinterlacer_mode` controls the function of the deinterlacer and takes the following values:
  - `0` - Output one frame for each frame (ex. 25i -> 25p).
  - `1` - Output one frame for each field (ex. 25i -> 50p).
  - `2` - Similar as `0`, but it skips the spatial interlacing check.
  - `3` - Similar to `1`, but it skips the spatial interlacing check.

  As default the mode is selected automatic to avoid frame duplication / drops. If the output framerate is less than 50% higher than the input framerate mode `0` is used, if higher `1` is used.
- `deinterlacer_parity` The picture field parity assumed for the input interlaced video. It accepts one of the following values:
  - `0`, tff - Assume the top field is first.
  - `1`, bff - Assume the bottom field is first.
  - `-1`, auto - Enable automatic detection of field parity.

  The default value is `-1` (auto). If the interlacing is unknown or the decoder does not export this information, top field first will be assumed.
- `deint` Specify which frames to deinterlace. Accepts one of the following values:
  - `0`, all - Deinterlace all frames.
  - `1`, interlaced - Only deinterlace frames marked as interlaced.

  The default value is `0` (all).

### Example:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
...
    <video>
...
      <setting>
        <key>deinterlacer</key>
        <value>advanced</value>
      </setting>
      <setting>
        <key>deinterlacer_parity</key>
        <value>1</value>
      </setting>
      <setting>
        <key>deinterlacer_mode</key>
        <value>1</value>
      </setting>
    </video>
```

```
    <metadata/>
</TranscodePresetDocument>
```

### scaling / cropping / rotating video

Used to set cropping and scaling parameters to the transcoder.

By default, the transcoder will attempt to maintain the display aspect ratio (DAR) of the cropped input. Use targetDAR to specify a different DAR to maintain.

The transcoder will typically try to adjust the pixel aspect ratio (PAR) so that the cropped picture ends up with the correct DAR. This minimizes the amount of processing required. Use pixelAspectRatio to set the PAR explicitly, in which case either width or height will be adjusted to maintain DAR. Use width and height to scale in those dimensions. If only one of them is set and PAR is set, the other one will be adjusted so the result matches the target DAR. If both are set and and PAR is set, the transcoder will take them as is.

Setting neither width nor height while PAR is set results in undefined behavior.

The transcoder will always double-check the resulting dimensions and PAR against the desired DAR. If there's a mismatch, the job will fail. If you want to force the transcoder to accept your settings, set targetDAR manually to the resulting DAR.

- `top`, `bottom`, `left`, `right` is used to specify cropping of the image, negative values will pad the picture with `padColor`.
- `rotate` - specifies how the image should be rotated, supported values are:
  - `right` - rotate the image -90 degrees
  - `left` - rotate the image 90 degrees
  - `upsidedown` - rotate the image 180 degrees
  - `autorotate` - rotate the image based on video metadata

### Example

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
...
    <video>
...
      <scaling>
        <width>1000</width>
        <height>2000</height>
        <top>-40</top>
        <bottom>-50</bottom>
        <left>-60</left>
        <right>-70</right>
        <padColor>#c0c0c0</padColor>
        <rotate>right</rotate>
        <targetDAR>
          <horizontal>1000</horizontal>
          <vertical>2000</vertical>
        </targetDAR>
      </scaling>
    </video>
    <metadata/>
</TranscodePresetDocument>
```

### 4.2.4 Custom settings

Some codecs support fine-grained custom settings. These settings are specified by adding `setting` element inside the `video` or `audio` element, or by adding a `muxerSetting`.

#### Common settings for video and image

##### thumbnailformat

By default, video thumbnails are JPEG and image thumbnails and PNG.

This can be changed with `thumbnailformat`. Valid values are `jpeg` and `png`.

#### Video-only settings

##### sceneChangeThreshold

Can be used to control GOP structure based on scene changes for `mpeg2video`. By default, GOPs are adjusted according to detected scene changes. Set to a very high number (1000000000) to disable scene change detection in order to get equal-sized GOPs.

##### noTimeCodeTrack

If true, do not write time code track. Primarily used for MP4 and MOV containers.

#### Image-only settings

##### colorspace

Sets the color space to specified value. Valid values are `CIELab`, `CMY`, `CMYK`, `Gray`, `HCL`, `HCLp`, `HSB`, `HSI`, `HSL`, `HSV`, `HWB`, `Lab`, `LCH`, `LCHab`, `LCHuv`, `LMS`, `Log`, `Luv`, `OHTA`, `Rec601Luma`, `Rec601YCbCr`, `Rec709Luma`, `Rec709YCbCr`, `RGB`, `scRGB`, `sRGB`, `Transparent`, `XYZ`, `YCbCr`, `YDbDr`, `YCC`, `YIQ`, `YPbPr`, `YUV`.

In addition, `colorspace` can be set as `demuxerSetting` as well. This works for PDF and PS files, and will cause PDF/PS parsing to generate result in the correct color space already when reading the file.

##### profile

Sets a profile. Profiles must be installed on transcoder node ( `/usr/share/color/icc`).

The transcoder now comes preinstalled with the most common profiles.

A special profile is added, `detect-xmp`. The profile will read the profile from XMP metadata and use that profile. An example, that will read the XMP profile and then set sRGB:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  ...
  <setting>
    <key>profile</key>
    <value>detect-xmp</value>
  </setting>
  <setting>
    <key>profile</key>
    <value>sRGB</value>
  </setting>
</TranscodePresetDocument>
```

### scaling algorithm

New in version 21.3.

It is possible to select which scaling algoritm that should be used when scaling the video.

Accepted values:

- `SWS_FAST_BILINEAR` - Bilinear scaling with some short cuts to give higher performance.
- `SWS_BILINEAR` - Bilinear scaling, uses a 2x2 environment of a pixel and then takes the average of these pixels to interpolate the new value.
- `SWS_BICUBIC` - Bicubic scaling, uses a 4x4 environment of a pixel, weighing the innermost pixels higher, and then takes the average to interpolate the new value.
- `SWS_X` - FFmpeg experimental algorithm.
- `SWS_POINT` - Fastest, uses closest point, gives a pixelated result when upscaling, might give ok result for down scaling
- `SWS_AREA` - Uses a mapping of source and destination pixels, averaging the source pixels with regards to the fraction of destination pixels that are covered.
- `SWS_BICUBLIN` - Uses bicubic scaling for luma values, and bilinear for croma values.
- `SWS_GAUSS` - Gaussian scaling, usually used for computer vision, not very useful for video.
- `SWS_SINC` - Uses higher-order polynomials and are therefore harder to compute than bicubic interpolation.
- `SWS_LANCZOS` - Resampling involves a sinc filter as well. It is more computationally expensive.
- `SWS_SPLICE` - Use higher-order polynomials and are therefore harder to compute than bicubic interpolation.

Changed in version 21.3.

The default value is `SWS_BICUBIC`, in older version SWS_FAST_BILINEAR was used.

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  ...
  <setting>
    <key>scaling_mode</key>
    <value>SWS_BICUBIC</value>
  </setting>
</TranscodePresetDocument>
```

### strip

If `true`, strip profile info. If `false`, do not strip profile.

### density

Set the density (resolution) of the image. The format is `xRes[ "x" yRes] WHITESPACE ( "dpi" | "dpcm" )`. If only one value is set, the same resolution is used for x and y. By specifying `dpi` or `dpcm`, resolution can explicitly be set to mean pixels per inch or pixels per centimeter, respectively.

### sharpen

If `true`, sharpens the image. May produce better results after scaling.

#### alpha extraction

New in version 5.0.

Accepted values:

- `keep_alpha` - The alpha value is kept and the color and luminance information is discarded.
- `discard_alpha` - The alpha information is discarded and the color and luminance information is kept.

Normally, if you have a source video with an alpha layer and transcode it to a format that doesn't support alpha, the alpha information is taken into account when calculating the color and luminance of the output video. By using one of the above values, you can filter out the alpha or color and luminance components of the pixels.

### Muxer settings

#### streamOrder

Controls in which order streams are numbered in the output. Comma separated list of `audio`, `video`, `subtitle`.

### Demuxer settings

#### extract closed captions

It is possible to extract EIA-608(also known as "line 21 captions") from source file and store it in item metadata by specifying `extract_closed_captions` = `true` in `demuxerSetting`.

From version 5.7, it is possible to extract all caption tracks, both EIA-608 and CEA-708, by specifying `extract_closed_captions` = `mcc` in `demuxerSetting`. The caption metadata vill contain a new field `stl_service` that shows what service it belongs to, `CC1` or `CC3` for EIA-608 and `S1` to `S64` for CEA-708.

#### Example: preset with extract_closed_captions=true

```
PUT /shape-tag/extract_CC
Content-Type: application/xml

<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mp4</format>
  <audio>
    <codec>aac</codec>
  </audio>
  <video>
    <codec>h264</codec>
  </video>
  <demuxerSetting>
    <key>extract_closed_captions</key>
    <value>true</value>
  </demuxerSetting>
</TranscodePresetDocument>
```

```
200 OK
```

#### Example: transcode using previously defined preset

```
PUT item/VX-82/transcode?tag=extract_CC
```

```
200 OK
```

When the job is finished item metadata will contain extracted closed caption *Subtitles* if the extraction was successful.

### Sequence render color settings

New in version 5.4.

A normal transcode operations only operates in YUV color space, and does not change the YUV values. However, a sequence rendering operation takes place in RGB color space, and gives the user more options with regards to output color space. There are four settings, added as video setting key-values pairs.

**Note:** Default values are unknown/unspecified. If you are unsure, use `bt709`.

### Color matrix

Key: `colorMatrix`

Available values:

- `gbr`
- `bt709`
- `unknown`
- `reserved`
- `fcc`
- `bt470bg`
- `smpte170m`
- `smpte240m`
- `ycgco`
- `bt2020nc`
- `bt2020c`
- `smpte2085`
- `chroma-derived-nc`
- `chroma-derived-c`
- `ictcp`

### Color transfer function

Key: `colorTransferFunction`

Available values:

- `reserved`
- `bt709`
- `unknown`
- `reserved`

- bt470m
- bt470bg
- smpte170m
- smpte240m
- linear
- log100
- log316
- iec61966-2-4
- bt1361e
- iec61966-2-1
- bt2020-10
- bt2020-12
- smpte2084
- smpte428
- arib-std-b67

### Color primaries

Key: colorPrimaries

Available values:

- reserved
- bt709
- unknown
- reserved
- bt470m
- bt470bg
- smpte170m
- smpte240m
- film
- bt2020
- smpte428
- smpte431
- smpte432
- ebu3213

#### RGB sample bit resolution

While the conversion between YUV and RGB is lossless and reversible in theory, round-off errors may occur. By setting the internal RGB sample width to 16, these errors are eliminated.

Key: `renderQuality`

Available values:

- `8` (default)
- `16`

## 4.3 Common presets

It is not always straightforward to construct a transcode preset that result in output with the desired format. Here are some guidelines for some of the most common formats.

### 4.3.1 H.264

The `codec` element should be set to `h264`. The default profile is Baseline. This can be overridden using the `preset` element. The following values are accepted:

- `baseline`
- `ipod`
- `main`
- `high`

There are also AVC-Intra specific profiles, see below.

#### Example

An MP4 using the Main profile:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mp4</format>
  <audio>
    <codec>mp3</codec>
    <framerate>
      <numerator>1</numerator>
      <denominator>44100</denominator>
    </framerate>
    <channel>0</channel>
    <channel>1</channel>
    <stream>2</stream>
  </audio>
  <video>
    <scaling>
      <width>1280</width>
      <height>720</height>
    </scaling>
    <codec>h264</codec>
    <bitrate>3000000</bitrate>
    <framerate>
      <numerator>1</numerator>
      <denominator>25</denominator>
    </framerate>
```

```
    <preset>main</preset>
  </video>
</TranscodePresetDocument>
```

New in version 5.0.

Depending on your license key Vidispine will use the H.264 encoder library from either MainConcept or Nablet. If your license allows for both, the Nablet version will be picked when using the *h264* codec tag. You can override this by using the vendor specific codec tags: *nablet_h264* and *mc_h264*.

### 4.3.2 AVC-Intra

To produce AVC-Intra output, the `preset` element should be set to `intra50` or `intra100` depending on desired output. Also add a setting of `codecTagString` to further specify the variant of AVC-Intra. The possible values are:

- `ai5p` – 50M 720p24/p30/p60
- `ai5q` – 50M 720p25/p50
- `ai56` – 50M 1080i60
- `ai55` – 50M 1080i50
- `ai53` – 50M 1080p24/p30
- `ai52` – 50M 1080p25
- `ai1p` – 100M 720p24/p30/p60
- `ai1q` – 100M 720p25/p50
- `ai16` – 100M 1080i60
- `ai15` – 100M 1080i50
- `ai13` – 100M 1080p24/p30
- `ai12` – 100M 1080p25

#### Example

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mov</format>
  <audio>
    <codec>pcm_s16le</codec>
    <framerate>
      <numerator>1</numerator>
      <denominator>48000</denominator>
    </framerate>
    <channel>0</channel>
    <channel>1</channel>
    <stream>2</stream>
  </audio>
  <video>
    <scaling>
      <width>1920</width>
      <height>1080</height>
    </scaling>
    <codec>h264</codec>
    <bitrate>100000000</bitrate>
    <framerate>
      <numerator>1</numerator>
```

```
      <denominator>25</denominator>
    </framerate>
    <gopSize>0</gopSize>
    <pixelFormat>yuv422p</pixelFormat>
    <preset>intra100</preset>
    <profile>CBR</profile>
    <setting>
      <key>codecTagString</key>
      <value>ai12</value>
    </setting>
  </video>
</TranscodePresetDocument>
```

### 4.3.3 ProRes

Set the `codec` element to `prores`. The `preset` element must also be set to one of the following values:

- `PR422HQ` – ProRes HQ
- `PR422` – ProRes 422
- `PR422LT` – ProRes LT
- `PR422Proxy` – ProRes Proxy
- `PR4444` – ProRes 4444
- `PR4444XQ` – ProRes 4444 XQ

The ProRes encoder will use the field-order information that Vidispine can read from the input file. In the case that Vidispine has the wrong information, you can override it by adding a `setting` key-value to the `video` element in the TranscodePresetDocument. The `key` should be `interlace_flag` and value one of:

- `progressive`
- `top_first`
- `bottom_first`

#### Example

ProRes 422 LT:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mov</format>
  <audio>
    <codec>pcm_s16le</codec>
    <framerate>
      <numerator>1</numerator>
      <denominator>48000</denominator>
    </framerate>
    <channel>0</channel>
    <channel>1</channel>
    <stream>2</stream>
  </audio>
  <video>
    <scaling>
      <width>1920</width>
      <height>1080</height>
    </scaling>
    <codec>prores</codec>
```

```
    <bitrate>85000000</bitrate>
    <preset>PR422LT</preset>
    <framerate>
      <numerator>1</numerator>
      <denominator>25</denominator>
    </framerate>
  </video>
</TranscodePresetDocument>
```

With `interlace_flag` set to `top_first`:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mov</format>
  <audio>
    <codec>pcm_s16le</codec>
    <framerate>
      <numerator>1</numerator>
      <denominator>48000</denominator>
    </framerate>
    <channel>0</channel>
    <channel>1</channel>
    <stream>2</stream>
  </audio>
  <video>
    <scaling>
      <width>1920</width>
      <height>1080</height>
    </scaling>
    <codec>prores</codec>
    <bitrate>85000000</bitrate>
    <framerate>
      <numerator>1</numerator>
      <denominator>25</denominator>
    </framerate>
    <preset>PR422LT</preset>
    <setting>
      <key>interlace_flag</key>
      <value>top_first</value>
    </setting>
  </video>
</TranscodePresetDocument>
```

### 4.3.4 XDCAM IMX-30/40/50

The `preset` element must be set to `imx30`, `imx40` or `imx50` depending on desired output. Also, a setting must be added specifying `codecTagString`. Accepted values are:

- `mx5p` – IMX-50
- `mx4p` – IMX-40
- `mx3p` – IMX-30

#### NTSC

To get NTSC output, there are a few changes that need to be made.

- The `framerate` should have a `numerator` of `1001` and a `denominator` of `30000`.

- The `scaling` element should have a `height` of `518`.
- Exchange the last letter of the `codedTagString` from `p` to `n` (i.e. `mx5p` to `mx5n`)

### Example

IMX-50:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mxf_d10</format>
  <audio>
    <codec>pcm_s24le</codec>
    <channel>0</channel>
    <channel>1</channel>
    <channel>2</channel>
    <channel>3</channel>
    <stream>4</stream>
  </audio>
  <video>
    <scaling>
      <width>720</width>
      <height>608</height>
      <top>-32</top>
    </scaling>
    <codec>mpeg2video</codec>
    <bitrate>50000000</bitrate>
    <framerate>
      <numerator>1</numerator>
      <denominator>25</denominator>
    </framerate>
    <displayWidth>
      <numerator>720</numerator>
      <denominator>1</denominator>
    </displayWidth>
    <displayHeight>
      <numerator>576</numerator>
      <denominator>1</denominator>
    </displayHeight>
    <displayXOffset>
      <numerator>0</numerator>
      <denominator>1</denominator>
    </displayXOffset>
    <displayYOffset>
      <numerator>32</numerator>
      <denominator>1</denominator>
    </displayYOffset>
    <containerSAR>
      <horizontal>64</horizontal>
      <vertical>45</vertical>
    </containerSAR>
    <gopSize>0</gopSize>
    <pixelFormat>yuv422p</pixelFormat>
    <preset>imx50</preset>
    <setting>
      <key>codecTagString</key>
      <value>mx5p</value>
    </setting>
  </video>
</TranscodePresetDocument>
```

### 4.3.5 XDCAM HD422

The `format` element must be set to `mxf_ffmpeg`. There are also some settings that must be added, see example below. The `codecTagString` setting should be one of the following values:

- `xd54` – 720p24 50Mb/s CBR
- `xd55` – 720p25 50Mb/s CBR
- `xd59` – 720p60 50Mb/s CBR
- `xd5a` – 720p50 50Mb/s CBR
- `xd5b` – 1080i60 50Mb/s CBR
- `xd5c` – 1080i50 50Mb/s CBR
- `xd5d` – 1080p24 50Mb/s CBR
- `xd5e` – 1080p25 50Mb/s CBR
- `xd5f` – 1080p30 50Mb/s CBR

#### NTSC

To get NTSC output, set the `framerate` to have a `numerator` of `1001` and a `denominator` of `30000`, and use the appropriate `codecTagString` from the list above.

#### Example

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mxf_ffmpeg</format>
  <audio>
    <codec>pcm_s24le</codec>
    <channel>0</channel>
    <channel>1</channel>
    <channel>2</channel>
    <channel>3</channel>
    <stream>1</stream>
    <stream>1</stream>
    <stream>1</stream>
    <stream>1</stream>
  </audio>
  <video>
    <scaling>
      <width>1920</width>
      <height>1080</height>
    </scaling>
    <codec>mpeg2video</codec>
    <bitrate>50000000</bitrate>
    <framerate>
      <numerator>1</numerator>
      <denominator>25</denominator>
    </framerate>
    <pixelFormat>yuv422p</pixelFormat>
    <setting>
      <key>flags</key>
      <value>+ildct+ilme</value>
    </setting>
    <setting>
      <key>top</key>
      <value>1</value>
```

```
    </setting>
    <setting>
      <key>dc</key>
      <value>10</value>
    </setting>
    <setting>
      <key>qmin</key>
      <value>1</value>
    </setting>
    <setting>
      <key>lmin</key>
      <value>1*QP2LAMBDA</value>
    </setting>
    <setting>
      <key>rc_max_vbv_use</key>
      <value>1</value>
    </setting>
    <setting>
      <key>rc_min_vbv_use</key>
      <value>1</value>
    </setting>
    <setting>
      <key>minrate</key>
      <value>50000k</value>
    </setting>
    <setting>
      <key>maxrate</key>
      <value>50000k</value>
    </setting>
    <setting>
      <key>bufsize</key>
      <value>36408333</value>
    </setting>
    <setting>
      <key>bf</key>
      <value>2</value>
    </setting>
    <setting>
      <key>codecTagString</key>
      <value>xd5c</value>
    </setting>
  </video>
</TranscodePresetDocument>
```

New in version 5.0.

Depending on your license key Vidispine will use the XDCamHD encoder library from either MainConcept or Nablet. If your license allows for both, the Nablet version will be picked when using the *mpeg2video* codec tag. You can override this by using the vendor specific codec tags: *nablet_mpeg2video* and *mc_mpeg2video*.

Valid preset tags to use in the shape-tags are, final XDCamHD profile will be determined of input framerate:

- xdcam_ex_1920
- xdcam_ex_1440
- xdcam_ex_1280
- xdcam_hd_420_1440

- xdcam_hd_420_1280
- xdcam_hd_422_1920
- xdcam_hd_422_1280

### 4.3.6 DV

For DVCAM, DVCPRO and DVCPRO50, `codec` should be set to `dvvideo`, for DVCPRO HD, it should be `dv_100`. To get 16x9 aspect ratio, `targetDAR` must be set (see example below). The value of `pixelFormat` determines whether the output will be DV, DVCPRO or DVCPRO50.

| Pixel format | Output |
|---|---|
| `yuv420p` | DVCAM |
| `yuv411p` | DVCPRO |
| `yuv422p` | DVCPRO50 |

#### NTSC

To get NTSC output the following changes should be made.

- The `framerate` should have a `numerator` of `1001` and a `denominator` of `30000`.
- The `scaling` should have a `height` of `480`.
- `codecTagString` should have a value of `dvpn`.

#### Example

16x9 DVCPRO:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>avi</format>
  <audio>
    <codec>pcm_s16le</codec>
    <framerate>
      <numerator>1</numerator>
      <denominator>48000</denominator>
    </framerate>
    <channel>0</channel>
    <channel>1</channel>
    <stream>2</stream>
  </audio>
  <video>
    <scaling>
      <width>720</width>
      <height>576</height>
      <targetDAR>
        <horizontal>16</horizontal>
        <vertical>9</vertical>
      </targetDAR>
    </scaling>
    <codec>dvvideo</codec>
    <bitrate>25000000</bitrate>
    <framerate>
      <numerator>1</numerator>
      <denominator>25</denominator>
    </framerate>
    <gopSize>0</gopSize>
    <pixelFormat>yuv411p</pixelFormat>
```

```
    <profile>CBR</profile>
    <setting>
      <key>codecTagString</key>
      <value>dvpp</value>
    </setting>
    <setting>
      <key>dtsmode</key>
      <value>pts</value>
    </setting>
  </video>
</TranscodePresetDocument>
```

### 4.3.7 DNxHD

The `codec` should be set to `dnxhd`.

#### Avid DNxHR/DNxHD

In version 4.13, the MainConcept codec is available for DNxHD encoding. Set `codec` to `mc_vc3`. Set the preset element to one of the following values.

| preset | width | height | interlace_flag | quality | Avid profile | Id |
|---|---|---|---|---|---|---|
| VC3_SQ_720p_TR | 960 | 720 | progressive | medium | Avid DNxHD 100 | 1258 |
| VC3_SQ_720p | 1280 | 720 | progressive | medium | Avid DNxHD 145 | 1252 |
| VC3_HQ_720p | 1280 | 720 | progressive | high | Avid DNxHD 220 | 1251 |
| VC3_HQX_720p | 1280 | 720 | progressive | high extended | Avid DNxHD 220x | 1250 |
| VC3_LB_1080p | 1920 | 1080 | progressive | low | Avid DNxHD 36 | 1253 |
| VC3_SQ_1080p_TR | 1440 | 1080 | progressive | medium | Avid DNxHD 100 | 1259 |
| VC3_SQ_1080p | 1920 | 1080 | progressive | medium | Avid DNxHD 145 | 1237 |
| VC3_HQ_1080p | 1920 | 1080 | progressive | high | Avid DNxHD 220 | 1238 |
| VC3_HQX_1080p | 1920 | 1080 | progressive | high extended | Avid DNxHD 220x | 1236 |
| VC3_444_1080p | 1920 | 1080 | progressive | RGB 4:4:4 | Avid DNxHD 444 | 1256 |
| VC3_SQ_1080i_TR | 1440 | 1080 | top_first | medium | Avid DNxHD 100 | 1243 |
| VC3_SQ_1080i | 1920 | 1080 | top_first | medium | Avid DNxHD 145 | 1244 |
| VC3_HQ_1080i | 1920 | 1080 | top_first | high | Avid DNxHD 220 | 1242 |
| VC3_HQX_1080i | 1920 | 1080 | top_first | high extended | Avid DNxHD 220x | 1241 |
| VC3_HQ_DCI_2K | 2048 | 1080 | progressive | high | | 1272 |
| VC3_HQX_DCI_2K | 2048 | 1080 | progressive | high extended | | 1271 |
| VC3_444_DCI_2K | 2048 | 1080 | progressive | RGB 4:4:4 | | 1270 |
| VC3_HQ_DCI_4K | 4096 | 2160 | progressive | high | | 1272 |
| VC3_HQX_DCI_4K | 4096 | 2160 | progressive | high extended | | 1271 |
| VC3_444_DCI_4K | 4096 | 2160 | progressive | RGB 4:4:4 | | 1270 |
| VC3_LB | any | any | progressive | low | Avid DNxHR LB | 1274 |
| VC3_SQ | any | any | progressive | medium | Avid DNxHR SQ | 1273 |
| VC3_HQ | any | any | progressive | high | Avid DNxHR HQ | 1272 |
| VC3_HQX | any | any | progressive | high extended | Avid DNxHR HQX | 1271 |
| VC3_444 | any | any | progressive | RGB 4:4:4 | Avid DNxHR 444 | 1270 |

Example:

```
<TranscodePresetDocument>
  <format>mxf</format>
  <audio>
      ...
  </audio>
```

```
  <video>
    <codec>mc_vc3</codec>
    <preset>VC3_HQ_1080p</preset>
    <setting>
        <key>interlace_flag</key>
        <value>progressive</value>
    </setting>
    <scaling>
        <width>1920</width>
        <height>1080</height>
    </scaling>
  </video>
</TranscodePresetDocument>
```

### 4.3.8 RED

Vidispine support RED as an input format so there is no special shape-tag settings that needs to be made. However, there are a few limitations and things to keep in mind.

#### Local file access

The transcoder needs to be able to read the RED file locally. Transcoder and Middleware needs to be running at the same machine.

#### Choosing an appropriate quality

Demuxing of RED material is a very computational demanding task. Normal RED footage has a resolution of 4K or 5K. Decoding such a frame in full resolution and quality is sometimes a bit overkill. That is, when creating a lowres file in 640x360 resolution you can save a lot of time by decoding the RED footage in a lower resolution.

You can specify what decoding/demuxing quality the transcoder should use by setting the `demuxerSetting` element in your shape-tag:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mp4</format>
  <audio>
      ...
  </audio>
  <video>
    ...
  </video>
  <demuxerSetting>
    <key>r3d_demuxer_quality</key>
    <value>full_premium</value>
  </demuxerSetting>
</TranscodePresetDocument>
```

Valid values for r3d_demuxer_quality is:

- `full_premium` - Full resolution and the best quality
- `half_premium` - Half of the width and height of the original resolution and the best quality
- `half_good` - Half of the width and height of the original resolution with good quality
- `quarter_good` - Quarter of the width and height of the original resolution with good quality
- `eight_good` - An eight of the width and height of the original resolution with good quality
- `sixteenth_good` - A sixteenth of the width and height of the original resolution with good quality

#### Multi-file RED clips

In case of multi-file RED clip the naming of the clips will be crucial. They should already be named (which they are as default):

```
<filename><index>.R3D
```

To preserve the filename of a RED file you can add a *filename script* to the storage where the RED files will be imported. For example:

```
PUT /API/storage/<storage-id>/metadata/filenameScript HTTP/1.1
Content-Type:text/plain

if (context.getExtension() != null && (context.getExtension() == "R3D" || context.
↪getExtension() == "r3d"))
    "VX-" + context.getOriginalFilename();
else
    context.getFileId() + "." + context.getExtension();
```

Then you need to import the clips into an *placeholder*, this way there will only be one transcoded file instead of X (X being the number of clips). For example:

```
POST /API/import/placeholder?container=0&video=2
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    <field>
      <name>title</name>
      <value>My placeholder for RED files</value>
    </field>
  </timespan>
</MetadataDocument>
```

```
POST /API/import/placeholder/<placeholder-id>/video?uri=file:/REDTEST_001.R3D&tag=mp4
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
  </timespan>
</MetadataDocument>
```

```
POST /API/import/placeholder/<placeholder-id>/video?uri=file:/REDTEST_002.R3D&tag=mp4
Content-Type: application/xml

MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
  </timespan>
</MetadataDocument>
```

### 4.3.9 AAC using Nablet

New in version 5.0.

The `codec` element should be set to `nablet_aac` and the encoder has support for up to 8 channels. The default profile is Low Complexity (LC). This can be overridden using the `settings` element with key `AAC_PROFILE`. The following values are accepted:

- AAC_MAIN
- AAC_LC
- AAC_SSR
- AAC_LTP

The stereo mode can be set using the settings element with the key AAC_STEREO_MODE. The following values are accepted, default is: AAC_LR_STEREO:

- AAC_MONO
- AAC_LR_STEREO
- AAC_MS_STEREO
- AAC_JOINT_STEREO

Depending on how you want to mux the output stream you can use two different output formats. The default is ADIF, which normally is used for MP4/MOV muxing. For MPEG transport stream muxing it is preferred to use the ADTS format. Use the settings element with the key AAC_OUTPUT_FORMAT. The following values are accepted:

- AF_ADIF
- AF_ADTS

### Example

An MP4 using the Main profile:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mp4</format>
  <audio>
    <codec>nablet_aac</codec>
    <framerate>
      <numerator>1</numerator>
      <denominator>44100</denominator>
    </framerate>
    <channel>0</channel>
    <channel>1</channel>
    <stream>2</stream>
    <setting>
      <key>AAC_OUTPUT_FORMAT</key>
      <value>AF_ADIF</value>
    </setting>
    <setting>
      <key>AAC_PROFILE</key>
      <value>AAC_MAIN</value>
    </setting>
    <setting>
      <key>AAC_STEREO_MODE</key>
      <value>AAC_JOINT_STEREO</value>
    </setting>
  </audio>
  <video>
    <scaling>
      <width>1280</width>
      <height>720</height>
    </scaling>
    <codec>h264</codec>
    <bitrate>3000000</bitrate>
```

```
    <framerate>
      <numerator>1</numerator>
      <denominator>25</denominator>
    </framerate>
    <preset>main</preset>
  </video>
</TranscodePresetDocument>
```

CHAPTER

FIVE

# STORAGES AND FILES

## 5.1 Storages

Storages are where Vidispine will store any files that are ingested/created in the system. All files on a storage location will get an entry in the Vidispine database, containing state, file size, hash etc. This is to keep track of any file changes.

For information about files in storage, see *Files*.

### 5.1.1 Storages

#### Storage types

A storage must be designated a type, based on what type of operations are to be performed on the contained files. Operations in this context are transcode, move, delete, and destination (that is, placing new files here).

**LOCAL** A Vidispine specific storage, suitable for all operations. Note that LOCAL doesn't necessarily imply that the storage is physically local. It should however be a dedicated Vidispine storage. That is, files on such storages should not be written to/deleted by any external application.

**SHARED** A storage shared with another application, Vidispine will not create new files, nor perform any write operations here.

**REMOTE** A storage on a remote computer, files should be copied to a local storage before used.

**EXTERNAL** A storage placeholder.

**ARCHIVE** A storage meant for archiving, needs a plugin bean or a JavaScript, described in more detail at *Archive Integration*.

**EXPORT** Files are not monitored, but copy operations to here will create a file entry in the database.

#### Storage states

Storages will have one of the following states:

**NONE** Not used.

**READY** Operating normally.

**OFFLINE** No available storage method could be reached.

**FAILED** Currently not used in Vidispine.

**DISABLED** Currently not used in Vidispine.

**EVACUATING** Storage is being evacuated.

**EVACUATED** Evacuating process finished.

For more information about storage evacuation, see section on *Evacuating storages*.

### Storage priority

New in version 4.17.

Storage priority can be set when creating a storage. If a shape has duplicate files on different storages, the file on the highest priority storage will be selected as the source of transcoder or transfer jobs

Example:

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <priority>HIGH</priority>
  <method>
  ...
  </method>
</StorageDocument>
```

Available priority values are: `HIGHEST`, `HIGH`, `MEDIUM`, `LOW`, `LOWEST`. Default priority of a storage is `MEDIUM`.

### Storage groups

Storages can be placed in named groups, called storage groups. These storage groups can then be used in *Storage rules* and *Quota rules*.

### Storage capacity

When a storage is created a capacity can be specified. This is the total number of bytes that is freely available on the storage. The free capacity is calculated as `total capacity -sum(file sizes in database list)`. Note that this means that the size of `MISSING` and `LOST` files are included in the used capacity. If you do not expect a file with these states to return, it is best to *delete the file entity using the API*.

### Auto-detecting the storage capacity

By setting the element `autoDetect` in the StorageDocument you can make Vidispine read the capacity from the file system. This only works if the storage has a storage method that points to the local file system, that is, a `file://` URI.

> **Warning:** Do not enable auto-detection for multiple storages located on the same device, as each storage will then have the capacity of the device. This means that storages may appear to have free space in Vidispine, when there is actually no space left on the device.

### Storage cleanup

If you have used storage rules to control the placement of files on storages then you may have noticed that files have been copied to the storages selected by the rules, but that files on the source storages have not been removed.

This is by design. Vidispine prefers to keep multiple copies of a file, and only remove the files when a storage is about to become full. The storage high and low watermarks control when files should start to be removed, and when enough files have been removed and storage cleanup should stop.

For example, for a 1 TB storage with a high watermark at 80% and a low watermark at 40%, Vidispine will keep adding files to the storage until the usage exceeds 800 GB. Once that happens cleanup would occur. Files that are deletable, that is, that have a copy on another storage and that is not required to exist according to the storage rules, will be deleted. Cleanup will stop once the usage has reached 400 GB or when there are no more deletable files.

If this behavior is not desirable, then there are two options.

1. Update the storage rules to specify where files should *not* exist, using the `not` element. For example, using `<not><any/></not>`.

```
<StorageRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <storageCount>1</storageCount>
  <storage>VX-122</storage>
  <not><any/></not>
</StorageRuleDocument>
```

2. Set the high watermark on the storage to 0%. Updating the storage rules is preferred as storage cleanup will be triggered continuously if the high watermark is set at a low level.

#### Evacuating storages

If you would like to delete a storage, but you still have files there which are connected to items, you can first trigger an evacuation of the storage. This will cause Vidispine to attempt to delete redundant files, or move files to other storages. Once the evacuation is complete, the storage will get the state `EVACUATED`.

### 5.1.2 Storage methods

Methods are the way Vidispine talks to the storage. Every method has a base URL. See *Storage method URIs* for the list of supported schemes.

Retrieve a storage to check its status. The storage `state` shows if the storage is accessible to Vidispine. If a storage is *not* accessible, then its state will be `OFFLINE`. Check the `failureMessage` in the storage methods to find out why. The failure message will be the error from when the last attempt to connect to the storage was made, and will be available even when the storage comes back online again. Compare `lastSuccess` to `lastFailure` to determine if the error message is current or not.

If multiple methods are defined for one storage, it is important, in order to avoid inconsistencies, that they all point to the same physical location. E.g. a storage might have one file system method, and one HTTP method. The HTTP URL must point to the same physical location as the file system method.

#### Storage method examples

Here are some examples of valid storage methods:

- `file:///mnt/vidistorage/`
- `ftp://vidispine:pA5sw0rd!?@10.85.0.10/storage/`
- `azure://:%2ZmFuODl0MGg0MmJ5ZnZuczc5YmhndjkrZThodnV5Ymhqb2lwbW9lcmN4c2Rmc2Q0NThmdjQ0Mzc`

#### Method types

Methods can also be of different type. By default, the type is empty. Only those methods (with empty types) are used by Vidispine when doing file operations, the other methods are ignored, but can be returned, for example when requesting URLs in search results.

Credentials are encrypted. This means that passwords cannot be viewed through the API/server logs.

#### Auto method types

One exception is method type `AUTO`, or any method type with prefix `AUTO-`. When a file URL is requested, with such method type, the a no-auth URL will be created (with the method URL as base).

If there is no `AUTO` method defined, but a file URL is requested with method type `AUTO`, an implicit one will be used automatically.

```
GET /item/VX-2406?content=uri&methodType=AUTO
Accept: application/xml
```

```
<ItemDocument xmlns="http://xml.vidispine.com/schema/vidispine" id="VX-2406">
  <files>
    <uri>http://vs.example.com:8089/APInoauth/storage/VX-1/file/VX-6537/0.
↪7354486788234469/VX-6537.mp4</uri>
    <uri>http://vs.example.com:8089/APInoauth/storage/VX-1/file/VX-6536/0.
↪7638025887084131/VX-6536.dv</uri>
  </files>
</ItemDocument>
```

The URL returned is only valid for the duration of *fileTempKeyDuration* minutes. The expiration timer is reset whenever the URL is used in a new operation (e.g. HEAD (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.4) or GET (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3)).

### AUTO-VSA method type

New in version 4.16.

When using URIs generated from the AUTO method type with a VSA storage, the files will be streamed from VSA through Vidispine server. Instead of that, the ÀUTO-VSA method type can be used to generate proxy URIs, which can later be used to generate noAuth URIs from the VSA on-demand.

The same Vidispine configuration property *fileTempKeyDuration* (default 10 minutes) is used to control the duration of both the proxy URI from the server and noAuth URI from the VSA.

**Example:**

First, generate a AUTO-VSA noauth URI:

```
GET /storage/file/VX-123?methodType=AUTO-VSA
Accept: application/xml
```

Response:

```
<FileDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-123</id>
  <path>demo.mov</path>
  <uri>
    http://localhost:8080/APInoauth/proxy/4e714b56-c3ab-49e9-b3f3-224aeaad7380?
↪redirect=true
  </uri>
  <state>CLOSED</state>
  ...
</FileDocument>
```

And then, ask VSA to generate a noauth URI.

```
GET http://localhost:8080/APInoauth/proxy/4e714b56-c3ab-49e9-b3f3-224aeaad7380?
↪redirect=true
```

Response:

```
HTTP/1.1 302 Found
Date: Thu, 20 Dec 2018 16:23:53 GMT
Accept-Ranges: bytes
Location: http://127.0.0.1:7090/4016eff6-5801-4ed2-a89d-518c9ee3b54a/demo.mov
Content-Length: 0
```

The URI in the Location header can be used to stream files from VSA directly.

The VSA noauth service will be running on port `7090` by default. And a `noAuthUri` property can be added to `agent.conf` to configure the noauth URI returned from the VSA.

For example:

```
noAuthUri=http://example.com:7090
```

### VSA related settings

#### Method metadata

In addition to select method types, method metadata can be given as instructions for the URI returned. Two metadata values are defined:

**format** Specifies if any special format of the URI should be returned. By default, the normal URI is returned. Two values are defined:

**SIGNED** Returns a `http` URI that points contains a signed URI directly to Azure or S3 storage. If a signed URI cannot be generated from the underlying (default) URI, no URI is returned.

**SIGNED-AUTO** As above, but if no URI can be generated, an `AUTO` URI (see above) is returned.

**expiration** Sets the expiration time of the signed URI, in minutes. If not specified, the expiration time is 60 minutes, unless `azureSasValidTime` is set.

**contentDisposition** Sets the Content-Disposition header for the signed URI. If not specified, the Content-Disposition header will be set to null.

**vsauri** Specifies if the VSA URI (schema `vxa`) should use UUID or name syntax. By default, UUID is used.

**UUID** Return URI with hostname being the UUID of the VSA.

**NAME** Return URI with hostname being the NAME of the VSA.

```
GET /item/VX-206?content=uri&methodMetadata=format=SIGNED-AUTO&
 ↪methodMetadata=contentDisposition=attachment%3b+filename%3dmyfile.mov
Accept: application/xml
```

```
<ItemDocument xmlns="http://xml.vidispine.com/schema/vidispine" id="VX-206">
  <files>
    <uri>https://vstest.s3.amazonaws.com/VX-362.mp4?Expires=1439545041&amp;
 ↪AWSAccessKeyId=AKIAJCCXQRY2MW4YQUVQ&amp;Signature=UcNdTIm1v1omM%2FaIGaYXf4QNfc%3D</
 ↪uri>
    <uri>http://vs.example.com:8089/APInoauth/storage/VX-1/file/VX-336/0.
 ↪7638025117084131/VX-336.dv</uri>
  </files>
</ItemDocument>
```

### Parent directory management

For local file systems (method is using a `file://` URI), Vidispine will by default remove empty parent directories when deleting the last file in the directory.

This can be controlled, either on system level or on storage level. If the storage metadata *`keepEmptyDirectories`* is set to true, empty directories are preserved in that storage. Likewise, if the configuration property *`keepEmptyDirectories`* is set to true, empty directories are preserved for all storages. Storage configuration overrules system configuration.

#### Storage scanning algorithm

By default, local file systems are scanned using what is called file *visitors*, which provides the best performance.

However, for some storages, especially mounted storages, ACLs on the file system may cause that algorithm to fail. By specifying the algorithm, if is possible to force VidiCore to use another algorithm.

This can be controlled, either on system level or on storage level, by the storage metadata *scanMethodAlgorithm*. Possible values are:

- `VISITOR` - use file visitors if possible, otherwise iterator. This is the default.
- `ITERATOR` - use file iterators
- `LEGACY`

### 5.1.3 Files

#### When are files scanned?

In order to discover changes made to files, or if any files have been removed/added, Vidispine will scan the storages periodically. It is possible to disable the scanning by not having any methods with `browse=true` on the storage. The scan interval is also configurable on a per storage basis by setting the `scanInterval` property. The value should be in seconds. Setting this to a higher value will lower the I/O load of the device, but any file changes will take longer to be discovered. This also means that file notifications for file changes or file creation will be triggered later for changes occurring outside of Vidispine's control.

You can force a rescan of a storage by calling *POST /storage/(storage-id)/rescan*. This will trigger an immediate rescan of a storage *if* the supervisor is idle. If a supervisor is already busy processing the files then you may notice that the rescan happens some time later.

#### Avoiding frequent scan of S3 storages

Scanning a S3 storage can be expensive both in terms of time and money. To make it cheaper to access a S3 bucket, you can configure Vidispine to poll Amazon SQS for S3 events (http://docs.aws.amazon.com/AmazonS3/latest/dev/NotificationHowTo.html).

See *S3 Event SQS Notifications* for more information.

#### File States

Files can be in one of the following states:

**NONE** Just created, not used.

**OPEN** Discovered or created, not yet marked as finished.

**CLOSED** File does no longer grow.

**UNKNOWN** The current state is not known.

**MISSING** File is missing from the file system/storage.

**LOST** File has been missing for a longer period. Candidate for restoration from archive.

**TO_APPEAR** File will appear on file system/storage, transfer subsystem or transcoder will create it.

**TO_BE_DELETED** The file is no longer in use, and will be deleted at the next clean-up sweep.

**BEING_READ** File is in use by transfer subsystem or transcoder.

**ARCHIVED** File is archived.

**AWAITING_SYNC** File will be synchronized by multi-site agent.

Vidispine will mark a file as `MISSING` when it is first detected that the file no longer exists on the storage. No action is taken for files that are missing. If the file does not appear within the time specified by *lostLimit*, then the file will be marked as `LOST`. Lost files will be restored from other copies if such exist.

### 5.1.4 Items and storages

By default, when creating a new file, Vidispine will choose the LOCAL storage with the highest free capacity. This can be changed in a few different ways:

- Setting the *defaultIngestStorage* configuration property.
- Supplying the *storageId* parameter on the import request.
- Using *Storage rules*.

### 5.1.5 File hashing

Vidispine will calculate a hash for all files in a storage. This is done by a background process, running continuously. Files are hashed one by one for performance reasons, so if a large number of files are added to the system in a short time span it might take some time for all hashes to be calculated. The default hashing algorithm is SHA-1. This can be changed by setting the configuration property *fileHashAlgorithm*. See below for a list of supported values.

#### Additional algorithms

Vidispine can be configured to calculate hashes using additional algorithms by setting the *additionalHash* metadata field on the storage. It should contain a comma separated list (no spaces) of algorithms. The supported algorithms are:

- MD2
- MD5
- SHA-1
- SHA-256
- SHA-384
- SHA-512

#### Manual hashing

Automatic background hashing can be disabled by setting the *hashMode* metadata field on the storage. A hash can then be set manually by calling *PUT {file-resource}/hash/(hash)*.

### 5.1.6 Throttling storage I/O

Vidispine will retrieve information about files on a storage at the configured scan intervals. If you find that the I/O on your local disk drives is high, even when no transfers or transcodes are being performed, then you can try rate limiting the stat calls performed by Vidispine. Do this by setting *statsPerSecond* or the configuration property *statsPerSecond* to a suitable limit. During the file system scan, Vidispine will typically perform one stat per file.

An easy way to check if rate limiting the stat calls will have any effect is to disable the storage supervisors in Vidispine. This can be done using `PUT /vidispine-service/service/StorageSupervisorServlet/disable`. Remember to enable the service afterwards or you will find that Vidispine no longer detects new files on the storages, among other things.

It could also be that it's the file hashing service that is the cause of the I/O. You should be able to tell which service is behind it by monitoring your disk devices. If there's a high read activity/a large amount of data read from a device then it could be the file hashing that's the cause. If the number of read operations per seconds is high then it's more likely the storage supervisor.

**Tip:** Use tools such as `htop`, `iotop`, `dstat` and `iostat` to monitor your systems and devices.

### 5.1.7 Throttling transfer to and from a storage

It is possible to specify a bandwidth on a storage or a specific storage method. This causes any file transfers involving the specified storage or storage method to be throttled. If multiple transfers take place concurrently, the total bandwidth will be allocated between the transfers. If a bandwidth is set on both the storage and its storage methods, the lowest applicable bandwidth will be used.

To set a bandwidth you can set the `bandwidth` element in the *StorageMethodDocument* when creating or updating a storage or storage method. The bandwidth is set in bytes per second.

#### Example

Updating a storage to set a bandwidth of 50,000,000 bytes per second.

```
PUT /storage/VX-2
Content-Type: application/xml

<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>LOCAL</type>
  <capacity>1000000000</capacity>
  <bandwidth>50000000</bandwidth>
</StorageDocument>
```

#### Example

Updating a storage method to set a bandwidth of 20,000,000 bytes per second.

```
PUT /storage/VX-2/method?uri=http://10.5.1.2/shared/&bandwidth=20000000
```

### 5.1.8 Temporary storages for transcoder output

The Vidispine transcoder requires that the destination (output) file can be partially updated. This is in order to be able to write header files after the essence has been written.

In previous versions, this is solved by the application server storing the intermediate result as a temporary file on the local file system (`/tmp`). This requires a lot of space on the application server.

With version 4.2.3, another strategy is available. Instead of storing the result as one file on the application server, several small files are stored directly on the destination file system as "segments". After the transcode has finished, the segments are merged. On S3 storage, this merging can be done with S3 object(s)-to-object copy.

Control of the segment file strategy is via the *useSegmentFiles* configuration property.

### 5.1.9 Storage credentials

Storage credentials can be specified in the storage URL, but can also be saved in an external location and referenced by an alias. This is configured in the server *configuration file*. Credentials can be stored in either:

- A Java Keystore (https://en.wikipedia.org/wiki/Keystore).
- HashiCorp Vault (https://www.vaultproject.io/).
- The local file system.

For example, a FTP storage could be configured either using `ftp://testuser:testpassword@ftp.example.com/`, or using `ftp://exampleftp@ftp.example.com/`; with `exampleftp` being an alias referencing the externally stored credentials.

### Java Keystore

A Java Keystore can be used to store private keys, for example, the private keys for a Google Cloud Platform service account.

Listing 5.1: server.yaml

```
secrets:
  keyStore:
    path: /etc/vidispine/server.keystore
    password: changeit
```

### Local file

For local file secret storage, the alias refers to the file under the configured secret path, containing the private key or username and password credentials.

- With private keys, the file should contain the private key as is.

In certain configurations where there is a directory present in the secrets path with the same alias, the private key should be stored under that directory as `private_key`.

- With username and password credentials, the file should be a directory, containing two files, `username` and `password`.
- To use a private key to authenticate a SFTP storage, the file should be a directory, containing the files `username`, `private_key` and `private_key_password`.

For example:

Listing 5.2: server.yaml

```
secrets:
  file:
    path: /etc/secrets/
```

```
$ mkdir -p /etc/secrets/exampleftp/
$ echo -n "testuser" > /etc/secrets/exampleftp/username
$ echo -n "testpassword" > /etc/secrets/exampleftp/password
$ echo -n "keypassphrase" > /etc/secrets/exampleftp/private_key_password
```

This could be one way to consume credentials from secrets in Kubernetes (http://kubernetes.io/docs/user-guide/secrets/), or similar services that expose secrets via the local file system.

### HashiCorp Vault

Using HashiCorp Vault the alias should match the name of a secret in Vault. Username and password credentials will be read from the keys `username` and `password`; private keys from the `private_key` key and passphrase to the private key from `private_key_password`.

For example:

Listing 5.3: server.yaml

```
secrets:
  vault:
    address: http://vault.example.com:8200
    token: 2262e94c-39c3-b9a8-605d-f0450dfc558b
    keyPrefix: secret/
```

The `keyPrefix` setting can be used to for example select the backend to use. For example, with Vault configured with a "generic" backend mounted at `secret/`:

```
$ vault mounts
Path        Type       Default TTL  Max TTL  Description
secret/     generic    system       system   generic secret storage
sys/        system     n/a          n/a      system endpoints used for control,␣
↪policy and debugging
```

```
$ vault write secret/exampleftp username=testuser password=testpassword
```

```
$ vault read secret/exampleftp
Key                 Value
---                 -----
refresh_interval    720h0m0s
password            testpassword
username            testuser
```

### 5.1.10 Storage method URIs

**Note:** Storage method URIs require URI escaping for all characters that are reserved in URIs.

The following URI schemes are defined.

**`file`**

**Syntax** `file:///{path}`

**Example** `file:///mnt/storage/`, `file:///C:/mystorage/`

**Note** The URI `file://mnt/storage/` is not valid! (But `file:/mnt/storage/` is.)

**`ftp`**

**Syntax** `ftp://{user}:{password}@{host}/{path}`

**Example** `ftp://johndoe:secr3t@example.com/mystorage/`

Add query parameter `passive=false` to force active mode. To set the client side ports used in active mode, set the configuration property `ftpActiveModePortRange`, the value should be a range, e.g. `42100-42200`.

To set the client IP used in active mode, set the configuration property `ftpActiveModeIp`.

New in version 4.17: For some servers using a basic implementation of ftp and which does not support some of the commands often found, e.g. listing a directory without having to step into it first, the query parameter `serverType=basic` can be used if issues with connecting and listing files are experienced. This will in some cases provide a better compatibility.

**sftp**

**Syntax** `sftp://{user}:{password}@{host}/{path}`

**Example** `sftp://johndoe:secr3t@example.com/mystorage/`

When using a private key to authenticate:

**Syntax** `sftp://{alias}@{host}/{path}`

**Example** `sftp://examplesftp@example.com/mystorage/`

**Note** Currently only PKCS#1 keys are supported; using vault or local secrets.

**http**

**Syntax** `http://{user}:{password}@{host}/{path}`

**Example** `http://johndoe:secr3t@example.com/mystorage/`

**Note** Requires WebDAV support in host.

**https**

**Syntax** `https://{user}:{password}@{host}/{path}`

**Example** `https://johndoe:secr3t@example.com/mystorage/`

**Note** Requires WebDAV support in host.

**omms**

**Syntax** `omms://{userId}:{userKey}@{hostList}/{clusterId}/{vaultId}/`

**Example** `omms://c2f6a2f4-6927-11e1-cc94-ab94bd11183f:some%20secret@10.0.0.3,10.0.0.4/425`

**Note** Object Matrix Matrix Store.

**s3**

**Syntax** `s3://{accessKey}:{secretKey}@{bucket}/{path}`

**Example** `s3://KDASODSALSDI8U:RxZYlu23NDSIN293002WdlNyq@mystore/storage1/`

If no access key is provided, then the credentials will be read from the `AwsCredentials.properties` file in the *`credentials directory`*, if one exists. Else, credentials will be read from the default locations used by the AWS SDK (http://docs.aws.amazon.com/AWSSdkDocsJava/latest/DeveloperGuide/java-dg-roles.html).

Valid S3 bucket names (https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/s3/model/Bucket.html) must agree with DNS requirements.

The following query parameters are supported:

**endpoint** The endpoint that the S3 requests will be sent to.

See Regions and Endpoints (http://docs.aws.amazon.com/general/latest/gr/rande.html) in the Amazon documentation for more information.

**region** The region that will be used in the S3 requests.

See Regions and Endpoints (http://docs.aws.amazon.com/general/latest/gr/rande.html) in the Amazon documentation for more information.

**signer** The algorithm to use to signing requests. Valid values include `S3SignerType` for AWS signature v2, and `AWSS3V4SignerType` for AWS signature v4.

**Default** Signature algorithm will be selected by region.

---

**Note:** For Version 4 Signature only regions (http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingAWSSDK.html#specify-signature-version) (Beijing and Frankfurt) to work, the endpoint or region parameter **must** be set. Example:

- `s3://frankfurt-bucket/?endpoint=s3.eu-central-1.amazonaws.com`
- `s3://frankfurt-bucket/?region=eu-central-1`

---

Storage method metadata keys can be used control the interaction with the storage.

**`storageClass`** The default Amazon S3 storage class that will be used for new files created on an Amazon S3 storage. Can be either `standard`, `infrequent` or `reduced`

**Default** `standard`

**`sseAlgorithm`** The encryption used to encrypt data on the server side. See Server-Side Encryption (http://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html). By default no encryption will be performed.

This sets the `x-amz-server-side-encryption` header on PUT Object S3 requests.

**Example** `AES256`

**`sseKeyId`** The encryption used to encrypt data on the server side. See Server-Side Encryption (http://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html). By default no encryption will be performed.

This sets the `x-amz-server-side-encryption-aws-kms-key-id` header on PUT Object S3 requests.

If the `sseAlgorithm` is present and has the value of `aws:kms`, this indicates the ID of the AWS Key Management Service (AWS KMS) master encryption key that was used for the object.

The KMS KEY you specify in the policy must use the `arn:aws:kms:region:acct-id:key/key-id` format.

**Example** `arn:aws:kms:us-west-2:360379543683:key/071a86ff-8881-4ba0-9230-95af6d01ca01`

**`accelerate`** Enable S3 Transfer Acceleration (http://docs.aws.amazon.com/AmazonS3/latest/dev/transfer-acceleration.html).

**Default** false

---

**Note:** For S3 Transfer Acceleration (http://docs.aws.amazon.com/AmazonS3/latest/dev/transfer-acceleration.html) to work, the endpoint or region parameter **must** be set. Also make sure that transfer acceleration is enabled on the bucket.

Other S3 compatible endpoints may not support transfer acceleration.

---

**`retrievalTier`** The default Glacier retrieval tier to use when restoring the file. Can be set to either `Expedited`, `Standard` or `Bulk`. See Restoring Archived Objects (http://docs.aws.amazon.com/AmazonS3/latest/dev/restoring-objects.html) for more information.

**`ssl`** Vidispine is by default using SSL when communicating with S3. Set to false to disable SSL support.

**Default** true

New in version 21.3.

**roleArn** The role ARN to try to assume to access the content of the bucket.

In order to be able to access buckets and content across accounts, it is now possible to supply a role ARN that VidiCore will try to assume to access the data.

**roleExternalId** The (optional) external id attached to the role specified as `roleArn`

**stsRegion** (optional) The region to where calls to assume role are made (AWS STS). This should be set to something as close to your system as possible to reduce latency and get better reponse times (example: `eu-west-1`, `us-east-2`).

---

**Note:**

- When a role is being assumed VidiCore will need to contact AWS Security Token Service (STS) in order to complete the request. Unless the system is running on EC2/ECS the best practice when using role ARN for S3 storages would be to make sure the `stsRegion` parameter is being used. If this is not supplied, VidiCore will take more time trying to figure out which region to call (see below).
- If no region is specified OR VidiCore is NOT running on EC2/ECS, VidiCore will fallback to the AWS default region which would be `us-west-2`. This is not recommended for optimal performance.

---

#### ds3

**Syntax** `ds3://{accessKey}:{secretKey}@{bucket}/{path}`

**Example** `ds3://KDASODSALSDI8U:RxZYlu23NDSIN2Nyq@bucketname/?endpoint=http://blackpearl-e`

**Note** Spectra BlackPearl Deep Storage Gateway.

The following query parameters are supported:

**endpoint** The endpoint of the BlackPearl service. This is mandatory.

**chunkReadyTimeout** The maximum time (in seconds) of waiting for BlackPearl to prepare the target data chunk, or an EOF will be returned.

**Default** `1800`

**checksumType** If set, a client-side checksum will be computed and sent to BlackPearl gateway for data integrity verification. Supported checksum types are: `md5`, `crc32` and `crc32c`.

**Default** Empty, no checksum will be sent.

#### azure

**Syntax** `azure://:{accessKey}@{accountName}/{containerName}`

**Example** `azure://:KLKau23dEE02WdlLiO@companyname/container1/`

#### gs

Google Cloud Storage (https://cloud.google.com/storage/).

Using a P12 *private key*:

**Syntax** `gs://{privateKeyAlias}@{bucket}/?project={project}&account={account}`

**Example** `gs://test-key-p12@test-bucket/?project=12345&account=67890`

Using a JSON private key:

**Syntax** `gs://{privateKeyAlias}@{bucket}/?project={project}`

**Example** `gs://test-key-json@test-bucket/?project=12345`

Using an OAuth2 access token:

**Syntax** `gs://:{accessToken}@{bucket}/?project={project}`

**Example** `gs://:abc123@test-bucket/?project=12345`

Using the credentials file specified in the GOOGLE_APPLICATION_CREDENTIALS environmental variable:

**Syntax** `gs://{bucket}/`

**Example** `gs://test-bucket/`

#### universal

A universal URI is used to create a universal storage method. A universal storage method does not have a root URI, instead all files contain their own absolute URI.

### 5.1.11 The `universal` storage method

A universal storage can be used to let Vidispine manage files which are not stored under a common root. Universal storages can be used like other storages, but there are certain differences. Before jumping to the differences, an example on how to use the storage:

#### Adding a universal storage

```
POST /storage
Content-Type: application/xml

<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>LOCAL</type>
  <capacity>150000000000</capacity>
  <method>
    <uri>universal:/</uri>
    <read>true</read>
    <write>true</write>
    <browse>true</browse>
  </method>
  <showImportables>true</showImportables>
</StorageDocument>
```

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>STORAGE-VX-722</id>
  <state>NONE</state>
  <type>LOCAL</type>
  <capacity>10000000000000</capacity>
  <freeCapacity>10000000000000</freeCapacity>
  <method>
    <id>STORAGEMETHOD-VX-728</id>
    <uri>universal:/</uri>
    <read>true</read>
    <write>true</write>
    <browse>true</browse>
    <type>NONE</type>
  </method>
  <metadata/>
  <lowWatermark>10000000000000</lowWatermark>
  <highWatermark>10000000000000</highWatermark>
```

```
  <showImportables>true</showImportables>
</StorageDocument>
```

### Adding a file

```
POST /storage/VX-722/file?path=file:///home/baz/vacation.mp4
```

```
<FileDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>FILE-VX-68264</id>
  <path>file:///home/baz/vacation.mp4</path>
  <uri>file:///home/baz/vacation.mp4</uri>
  <state>OPEN</state>
  <size>-1</size>
  <timestamp>2017-05-11T13:37:46.737+02:00</timestamp>
  <refreshFlag>1</refreshFlag>
  <storage>STORAGE-VX-722</storage>
  <metadata/>
</FileDocument>
```

After scanning, the metadata and hash checksum of the file will be updated.

### Adding and importing a file

A file can be registered to a universal storage with its original URI, and imported at the same time:

```
POST /storage/VX-722/file/import?path=https://www.vidispine.com/wp-content/themes/
↪vidispine/assets/image/vidispine-logo-small.png
```

The HTTPS URI in the request will be the actual source of the `original` shape of the item created.

Compared with a regular import request:

```
POST /import?uri=https://www.vidispine.com/wp-content/themes/vidispine/assets/image/
↪vidispine-logo-small.png
```

The source file will be copied to a *Vidispine managed storage*. The newly copied file will be the file that makes up the `original` shape of the item. The HTTPS URI is then no longer used after the import.

### Differences to regular methods

- New files are not discovered by a universal method. For new files to be registered, an API call has to be done. However, Vidispine will detect when files have changed or been deleted.
- Files can be written to a universal storage. However, it requires that either a full URI is given as API input, or returned by a file name script. Example for copying a file:

  ```
  POST /storage/file/VX-4448/storage/VX-722?filename=file:///tmp/somenewfile.txt
  ```

- There is no capacity detection.
- Scanning can be slower than for regular storages. The universal URI is not meant to be used to thousands of files in one file system. Then it is better to use the regular URI, and reference files by their relative paths.

## 5.2 Automatic import

A storage can be configured to automatically import new files/image sequences that are detected. Auto-import rules define what transcodes that should be performed as well as what metadata to be used if none can be found. Metadata

can automatically be found if it shares the same filename and has the extension `.xml`, for example `video.avi` and `video.xml`.

Auto-import rules can also use *Import settings* to set up access control lists by setting the optional `settingsId` element.

### 5.2.1 Import using a specific transcoder resource

To use a specific transcoder resource during auto import, specify its resource id in the auto import rule using the `resourceId` element:

```
PUT /storage/VX-2/auto-import
Content-Type: application/xml

<AutoImportRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <tag>myflvtag</tag>
  <resourceId>VX-1</resourceId>
</AutoImportRuleDocument>
```

You may also specify which transcoder resource to use by setting the *defaultTranscoder* configuration property.

### 5.2.2 Setting a user for jobs started as a result of an auto import rule

The default behavior is that jobs started from an auto import rule will not have a user set. This can be changed by setting the `user` element in the rule:

```
PUT /storage/VX-2/auto-import
Content-Type: application/xml

<AutoImportRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <tag>myflvtag</tag>
  <user>angela@company.com</user>
</AutoImportRuleDocument>
```

### 5.2.3 Importing with a metadata file of an external format

Vidispine also supports auto imports with a metadata XML file that is of a different format than the native Vidispine MetadataDocument. This is achieved by associating a *Metadata projections* (XSLT transformation) with the auto import rule. First, create the projection, then set the auto import rule:

```
PUT /storage/VX-2/auto-import
Content-Type: application/xml

<AutoImportRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <tag>myflvtag</tag>
  <projection>myProjection</projection>
</AutoImportRuleDocument>
```

Where the `projection` element contains a *Projection id*.

Any auto imports from this storage will then first transform the supplied XML file using the specified projection.

### 5.2.4 Disable automatic import rules

Auto-import rules can be disabled by setting the `enabled` field to `false`, or by using the *disable rule request*. When a rule is disabled then no new auto-import jobs will be created for new files discovered on the storage.

Jobs will be created once the rule is enabled again. By default rules are enabled.

#### Example

If the admin only wants the user to be able to import during specific hours a program like cron (https://help.ubuntu.com/community/CronHowto) could be used:

```
0 0 * * * curl -X PUT -uadmin:admin http://localhost:8080/API/storage/VX-1/auto-
↪import/enable
0 5 * * * curl -X PUT -uadmin:admin http://localhost:8080/API/storage/VX-1/auto-
↪import/disable
```

### 5.2.5 Sidecar auto import

If auto-import is enabled, sidecar files are by default identified by file extension and imported as metadata to files with matching filenames. See *Importing sidecar files* for the supported sidecar formats which are automatically identified. The AutoImportRuleDocument contains two fields determining how sidecar are handled by auto-import:

**ignoreSidecarImport**
: False by default. If set to true, files with a sidecar file extension are not imported in any way.

**disabledSidecarExtensions**
: Extensions can be specified here to be excluded from being treated as sidecar files, and may instead be imported as new items.

  **Since** 4.16

If there are several files in a storage with the same file name (prefix) with different extensions, only one will be imported as an item and others will be either imported as sidecars or ignored. This can be changed by setting the system-wide configuration property `groupImportableFiles` to false. Multiple files with the same prefix can then be auto-imported as individual items.

Multiple sidecar files can be imported to the same item. If *`groupImportableFiles`* is false, one sidecar file may be auto-imported multiple times as metadata for different items with the same file prefix.

### 5.2.6 Title as metadata

The AutoImportRuleDocument contains a field `fileNameAsTitle`. Setting this property to `true` means that the "title" fields of all single files imported form this storage will be set to their file names.

### 5.2.7 Applying file name filters to auto import rules

There are two kinds of filename filters that can be applied to auto import rules:

**Exclusion filters** Used to exclude files from being auto imported. This can be useful when the OS creates files automatically, e.g. `Thumbs.db` on Windows or `.DS_Store` files on Mac OS. Note that the expression must match the entire path, not only a part of the path.

**Shape tag filters** These can be used to transcode the imported file using a specific shape tag when a file name follows a certain pattern. You might want files ending in `.tiff` to be transcoded using the tag `lowimage` for example.

The filters are specified in the XML document you use to create/update the auto import rule.

#### Example

```
<AutoImportRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <metadata>
    <timespan start="-INF" end="+INF">
      <field>
        <name>title</name>
        <value>This is an auto-imported item.</value>
```

```
      </field>
    </timespan>
  </metadata>
  <tag>generictag</tag>
  <excludeFilter>
    <pattern>.*\.DS_Store</pattern>
  <excludeFilter>
  <shapeTagFilter>
    <pattern>.*\.tiff</pattern>
    <tag>lowimage</tag>
  </shapeTagFilter>
  <shapeTagFilter>
    <pattern>.*\.mxf</pattern>
    <tag>lowvideo</tag>
  </shapeTagFilter>
</AutoImportRuleDocument>
```

This rule will exclude any file ending with `.DS_Store`. Any files ending with `.tiff` will be imported with the shape tag `lowimage`, and any files ending in `.mxf` will be imported with the shape tag `lowvideo`. All files will be imported with the shape tag `generictag`.

### 5.2.8 Auto import of image sequences

Deprecated since version 4.6: Define a *sequence pattern on the storage* and use an auto-import rule without a sequence definition instead.

Image sequences can be auto detected and imported if their file names match the predefined regex in AutoImportRuleDocument. The elements in the document are:

**`fileSequence`** Defines the file name pattern, and it is **mandatory**.

**`sequenceMetadata`** Defines the metadata file name pattern.

**`idGroup`** The matching group in the regex should be used as the id of the file sequence.

**`numGroup`** The matching group in the regex that should represent the position of a file in a sequence.

**`timeout`** A sequence is considered as completed after a certain timeout (in seconds). The default timeout is 60 seconds.

#### Example:

```
<AutoImportRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
        <tag>mp4</tag>
        <metadata>
                <timespan end="+INF" start="-INF">
                        <field>
                                <name>title</name>
                                <value>auto-imported item.</value>
                        </field>
                </timespan>
        </metadata>
        <sequenceDefinition>
                <sequenceMetadata>
                        <regex>(.*)-metadata.xml</regex>
                        <idGroup>1</idGroup>
                </sequenceMetadata>

                <fileSequence>
```

```
                        <regex>(.*)-([0-9]+).(dpx|tga|png|jpg)</regex>
                        <idGroup>1</idGroup>
                        <numGroup>2</numGroup>
                        <timeout>10</timeout>
                        <!-- seconds-->
                </fileSequence>
        </sequenceDefinition>
</AutoImportRuleDocument>
```

Given a storage with the above import rule, with the files:

```
foo-metadata.xml
foo-001.dpx
foo-002.dpx
foo-002.dpx
```

Then these would be recognized as a sequence `foo` with `foo-metadata.xml` as the metadata.

## 5.3 Storage rules

Storage rules are a way of controlling the availability of files. The rules describe where files of different types are stored. Settings include a minimum number of storages, specific storages and priorities for how suited a storage is for a particular type. A rule can be applied on a specific item, collection, library or shape tag. To further filter which shapes that the rules applies to, a shape tag can be set. Files can be named using *storage name rules*.

A storage rule can also describe where files should **not** be stored, in which case files will be eagerly removed. The default is otherwise to start removing files once the high watermark on a storage has been reached. A rule can specify specific storages or storage groups, or that all other storages should be excluded by using the "all" qualifier.

> **Warning:** Negative rules do not work for storages of type ARCHIVE

Storage rules on collections will also be inherited to items in sub-collections. See *Inherited rule example*.

Files can also be *load balanced* across multiple storages.

### 5.3.1 Resolving storage rules

If a minimum number of storages has been set and an insufficient amount of specific storages are given, priorities are used to pick a suitable storage. The different priority criteria can be seen in the table below. The criteria type is given together with an integer describing its priority, where a lower number means that it is more important than an entry with a higher number.

| Type | Description |
|---|---|
| bandwidth | Prioritizes bandwidth. |
| capacity | Prioritizes free available space. |

#### Which rules apply?

Certain rules takes precedence over other rules. There are three things that factors into this decision process (ordered according to their importance):

1. The precedence given to the rule.
2. The type of the entity the rule is applied to.
3. Whether the rule is set to a certain shape tag or not.

Below a table of available precedence values can be seen, ordered from most important to least important.

| Name |
|---|
| HIGHEST |
| HIGH |
| MEDIUM (default value) |
| LOW |
| LOWEST |

Below a table of the difference entity types can be seen, ordered from most important to least important.

| Name |
|---|
| ITEM |
| COLLECTION |
| LIBRARY |
| GENERIC (the type used if set directly on a shape tag) |

So for example a rule with the precedence value HIGHEST, that is applied to a certain shape tag on an item will always take precedence over any other rule.

#### How are storage rules applied?

Since a shape can have 0 or more shape tags, there can be some ambiguity between the rules. Below a basic algorithm, that describes how the rules are applied, can be seen.

1. Start out with an empty set of storages, *S*.
2. Add all storages, given in the specific rules, to *S*.
3. If *S* is empty, add in storages specified in the generic rule.
4. Set the minimum required storages, *n*, to equal the highest number specified in the specific rules and the generic rule.
5. If the size of *S* is less than *n*:
   (a) Retrieve the priorities from one of the specific rules.
   (b) If no specific rule specified any priorities, use the generic rule.
   (c) If the generic rule did not specify any priorities, use some system default priorities.
   (d) Attempt to fill *S* using the priorities.

### 5.3.2 Examples

#### Simple rule example

Setting a simple rule on a item, dictating that the item's original shape should exist on at least two storages, and one of them must be storage VX-3

```
PUT /item/VX-28/storage-rule/original
Content-type: application/xml
```

```
<StorageRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <storageCount>2</storageCount>
  <storage>VX-3</storage>
</StorageRuleDocument>
```

### Negative rule example

Setting a simple rule on a item, dictating that the item's original shape should exist on at least two storages, and one of them must be storage VX-3, and it must not exist on storage VX-2.

```
PUT /item/VX-28/storage-rule/original
Content-type: application/xml
```

```
<StorageRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <storageCount>2</storageCount>
  <storage>VX-3</storage>
  <not>
    <storage>VX-2</storage>
  </not>
</StorageRuleDocument>
```

### Load balancing example

The `pool` element can be used to specify the storages that files could be stored on. This can be used to spread files across multiple selected storages.

For example, to specify that a file should exist on at least two of storages VX-1, VX-2 and VX-3:

```
<StorageRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <storageCount>2</storageCount>
  <pool>
    <storage>VX-1</storage>
    <storage>VX-2</storage>
    <storage>VX-3</storage>
  </pool>
</StorageRuleDocument>
```

Or to have one copy on S3 (storage VX-1 in this example) and one copy on a local storage:

```
<StorageRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <storageCount>2</storageCount>
  <storage>VX-1</storage>
  <pool>
    <group>local-storage-pool</group>
  </pool>
</StorageRuleDocument>
```

### Inherited rule example

Storage rules on collections by default only applies to the items in the collection and does *not* apply for items that exist in any sub-collections.

To change so that a collection storage rule applies to all items in it and all items in any sub-collections, recursively, use:

```
<StorageRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <storageCount>1</storageCount>
  <inherited>true</inherited>
  <storage>VX-1</storage>
</StorageRuleDocument>
```

## 5.4 Filenames

By default, Vidispine names new files according to **site-id**–**number****‘‘.‘‘****extension**, all in one folder. This pattern can be overridden. This section describes three very different ways.

### 5.4.1 Using a tree structure for files

Putting all files in the same directory of a storage can cause degraded performance on some file systems. By setting the configuration property `fileHierarchy`, the naming convention is changed to **site-id** – **number1** / **number2** . **extension**. The number set in `fileHierarchy` controls the size of **number2**. Example:

| `fileHierarchy` not set, or 0 | `fileHierarchy` =100 | `fileHierarchy` =1000 |
|---|---|---|
| `VX-7.mp4` | `VX-0/07.mp4` | `VX-0/007.mp4` |
| `VX-47232.mp4` | `VX-472/32.mp4` | `VX-47/232.mp4` |

Note that the splitting into subdirectories is currently only done in one level, so no `VX-4/72/32.mp4`.

The configuration property may be changed at any time, but old files will not be renamed.

### 5.4.2 Storage name rules

A storage name rule dictates the filename that the file of a particular shape should have on a certain storage. Note that these rules doesn't make sure a file is actually located on a storage, it just says what filename a file should have **if** it is located on that storage. Storage name rules are often used together with *storage rules*

### 5.4.3 Naming files on storage

The default naming convention of can be overridden on a per-storage basis by associating a JavaScript script to the storage.

The script will be invoked whenever a file needs to be created on the storage.

#### Setting the script

The JavaScript is stored as metadata `filenameScript` to the storage. That is, the code is set using `PUT /storage/{storage-id}/metadata/filenameScript`.

If using curl, use `--data-binary` instead of `-d` to make sure all new-line characters are kept.

#### Input

In the execution context of the script, there is a variable named `context`, which has the following functions:

`context.`**`getShape`**`()`
: Returns a ShapeType (see Vidispine XSDs) object.

  For example, to get the essence version, use `context.getShape().getEssenceVersion()`. Can return `null`.

`context.`**`getJobMetadata`**`()`
: Returns a `java.util.Map<String,String>`. Can be `null`.

`context.`**`getItem`**`()`
: Returns an ItemType, which is the same output as `GET /item/{item-id}?content=metadata,shape,access,exte`
  Can return `null`.

`context.`**`getStorage`**`()`
: Returns a StorageType.

`context.getComponent()`
: Returns a ComponentType. Can return `null`.

`context.getExtension()`
: Returns the suggested extension for the file. Can return `null`.

`context.getFileId()`
: Returns the file id of the file to be created.

`context.getTags()`
: Returns a `java.util.Collection<String>` of the shape tags of the shape the file belongs to.

`context.getOriginalFilename()`
: Returns the original filename that was used when item was imported.

`context.getOriginalComponentFilename()`
: Returns the original filename that was used when component was created.

`context.getChannel()`
: Most of the time this will return null, except when you want to *split audio channels to separate files*.

`context.getJobId()`
: Returns the job id.

`context.getJobType()`
: Returns the job type.

#### Output

The script should return (last value) the file name of the file.

#### Existing file names

If the suggested file name is already in use on the Storage, the script will be called again, up to 10 times. The new invocations will run in the same context as the previous, so it is possible to store information, e.g. sequence numbers, to not repeat the same file name.

#### Example

```
var l = "foobar-"+context.getStorage().getId()+"/"+context.getFileId();
if (context.getExtension() != null)
    l += "."+context.getExtension();
```

## 5.5 Image sequences

Image sequences can be imported into Vidispine just like any other video file.

### 5.5.1 Overview

An image sequence is made up of multiple sequentially numbered images files. Typically each file represents a video frame.

All files in a sequence are represented using a single file in Vidispine. Such files have a type of `FILE_SEQUENCE`.

```
<FileDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-5591787</id>
  <path>VX-5591787//.jpg</path>
  <uri>file:/srv/media/VX-5591787/*.jpg#file=0-49</uri>
```

```
  <state>CLOSED</state>
  <size>-1</size>
  <timestamp>2016-06-23T14:50:58.129+02:00</timestamp>
  <refreshFlag>1</refreshFlag>
  <storage>VX-1</storage>
  <metadata/>
  <range start="0" count="50"/>
  <type>FILE_SEQUENCE</type>
</FileDocument>
```

### 5.5.2 Importing image sequences

Image sequences can be imported using a *sequence URI*. For example, to import an image sequence whose sequence numbers are zero padded:

```
POST API/import?uri=file:///srv/data/take1/*.png#file=00000-15000
```

```
<JobDocument xmlns:ns0="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-127247</jobId>
  <user>admin</user>
  <started>2016-06-23T13:02:04.811Z</started>
  <status>READY</status>
  <type>PLACEHOLDER_IMPORT</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

### 5.5.3 Detection of image sequences

To have image sequences detected on a storage, the storage must first be configured with one or more *sequence patterns*. Otherwise, individual files in the sequence will appear as separate files.

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-1879</id>
  <state>NONE</state>
  <type>LOCAL</type>
  <capacity>16851361792</capacity>
  <freeCapacity>16763961344</freeCapacity>
  <timestamp>2016-06-23T15:04:02.873+02:00</timestamp>
  <method>
    <id>VX-1829</id>
    <uri>file:///srv/media/</uri>
    <read>true</read>
    <write>true</write>
    <browse>true</browse>
    <lastSuccess>2016-06-23T15:04:02.878+02:00</lastSuccess>
    <type>NONE</type>
  </method>
  <sequence>
    <regex>.*-(\d+).png</regex>
  </sequence>
</StorageDocument>
```

Files that match the pattern and that have the same sequence key will appear as a single file with type `FILE_SEQUENCE`. These files can then be imported directly from the storage just like any other files.

```
POST /storage/file/VX-46264/import
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    <!-- any metadata to add to the item -->
  </timespan>
</MetadataDocument>
```

### 5.5.4 Sequence URIs

Any URI containing a file fragment is considered as a sequence URI, that is, as a URI that identifies multiple files in a sequence. A sequence URI should contain a wildcard character that identifies the location of the sequence number.

Syntax: `file=[wildcard:][start[-end]][,...]`

- The default wildcard character is `*`. If the wildcard character is already present in the file name or path, then another wildcard character should be selected and specified in the file fragment.
- The default starting index is 0.
- The ending index is inclusive.
- When importing sequences using the API, only the last range in the file fragment may be open. That is, `file=0-10,40-` is supported, but `file=0-10,40-,10-20` is not.

Example:

- `file:///srv/media/inside-tardis/*.jpg#file=00000-90000`
- `file:///srv/media/inside-tardis/X.jpg#file=X:00000-90000`
- `http://media.example.com/?id=take1&num=*#file=0-`

### 5.5.5 Sequence patterns

The sequence pattern element contains a `<regex>`, and a optional `<numGroup>`.

The value of the `<regex>` should be a valid java regex string with some restrictions:

- There needs to be a `\d+`, which would match multiple digits. The matching digits will be used as the sequence number. The other parts will become the sequence key, which will be used to identify the sequence.

  **Important:** A sequence number can only contain one or multiple digits. Only numbers and zero padded numbers are supported

- `\d+` needs to be in a capturing groups, if there are other groups. And in this case, `<numGroup>` needs to be set.

The value of `<numGroup>` is the index of the sequence number capturing group.

Some Examples:

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-1879</id>
  ...
  <sequence>
    <regex>.*-(\d+).png</regex>
    <numGroup>1</numGroup>
```

```
  </sequence>
</StorageDocument>
```

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-1879</id>
  ...
  <sequence>
    <regex>(.*)-(\d+).png</regex>
    <numGroup>2</numGroup>
  </sequence>
</StorageDocument>
```

## 5.6 URI's, URL's, and Special Characters

### 5.6.1 File paths

There are a number of characters that have special uses in various file systems.

#### Characters not allowed in path segments (directory names, file names)

- U+0000 - U+001F (including TAB, CR, NL)
- U+002F ( / )
- U+005C ( \ )

While technically possible to use in path segments on various file systems, it is not possible to use these characters in Vidispine path names.

#### Characters not supported on certain platforms

- U+007F ( DEL )
- U+003F ( ? )
- U+002A ( * )
- U+0024 ( $ )
- U+003A ( : )
- Paths that are MS-DOS device names ( LPT1, etc)
- U+D800 - U+10FFFF

These characters may or may not work, depending on operating system and Java version. It is strongly suggested that they are not used.

### 5.6.2 API calls

In calls to the Vidispine API, the following rules apply:

- Path segments are encoded using RFC3986 (http://www.ietf.org/rfc/rfc3986.txt).
  - Non-ASCII characters are encoded in UTF-8, and do not have to be percent encoded.
  - Percent encoding. Particularly space is encoded as %20 (not +, so Java's URLEncoder is not the right tool!)
- Non-ASCII characters are encoded in UTF-8, and do not have to be percent encoded

- Percent encoding. Particularly space is encoded as %20 (not +, so Java's URLEncoder is not the right tool!)
- Query parameter values are encoded using RFC2396 (http://www.ietf.org/rfc/rfc2396.txt)
  - Non-ASCII characters need to be percent encoded.
  - Space can be encoded as + (or %20).
- Non-ASCII characters need to be percent encoded
- Space can be encoded as + (or %20)
- URIs in XML documents need to be quoted according to XML, e.g. & for &.

**Note:** As a consequence, path that are used as query parameters (e.g. the URL parameter in imports), need first to be encoded as a URI, then encoded as a URL query parameter.

### Example 1

Path: `/tmp/my movie.dv`

As a URI: `file:/tmp/my%20movie.dv`

As a URL parameter for import: `http://localhost:8080/API/import?URL=file%3A%2Ftmp%2Fmy%2520movie.dv` (see below)

Note that the space has to be quoted twice. First to %20 in the URI, than the percent sign in %20 have to be quoted to %2520.

### Example 2

Path: `/tmp/tête-à-tête.dv`

As a URI: `file:/t%C3%AAte-%C3%A0-t%C3%AAte.dv` (UTF-8 is used for the special characters, then percent encoded) (Optionally: `file:/tête-à-tête.dv` )

As a URL parameter for import: `http://localhost:8080/API/import?URL=t%25C3%25AAte-%25C3%25A0-t%25C3%`

### Code example

The following Java code, using Jersey's UriBuilder, shows how to generate valid API calls:

```
String path = "/tmp/tête-à-tête.dv";
URI uri = new File(path).toURI();
URI callUri = UriBuilder.fromUri("http://localhost:8080/API/import").queryParam("uri",
↪ "{uri}").build(uri);
```

**Warning:** In previous versions of Vidispine, the following call was accepted: `http://localhost:8080/API/import?URL=file:/tmp/my+movie.dv`. However, this is not valid, as the actual value of the parameter is then `file:/tmp/my movie.dv` , which is not a valid URI. (However, `http://localhost:8080/API/import?URL=file:/tmp/my%2520movie.dv` is valid.)

**See also:**

- The URLEncode and URLDecode Page (http://www.albionresearch.com/misc/urlencode.php)

CHAPTER

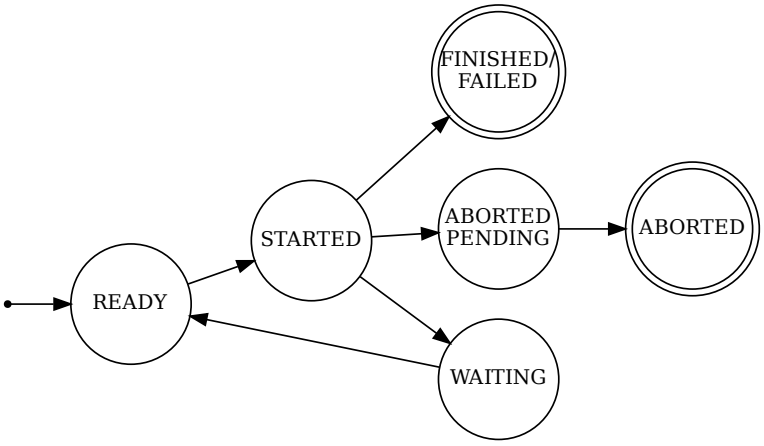
SIX

# JOBS AND TASK DEFINITIONS

## 6.1 Jobs

Jobs make up the long running tasks in Vidispine. They are created in response to requests that would otherwise not be able to respond in time, such as import, export and transcode requests.

The actions performed by a job is determined by its type. Bound to the type are a number of steps, or tasks, defined by the *task definitions*. The tasks form a graph, and typically execute in sequence, but it is also possible for tasks to start in parallel. This happens for example when importing and transcoding a growing file. The transfer step will initiate the transfer and then trigger the transcode step to start once enough data (the header) from the file has been transferred.




The states of a job are illustrated above. See below for a full description of the *states* and of the *job step states*.

### 6.1.1 Creating jobs

Create jobs by making requests to other RESTful resources:

| Job type | Relevant documentation |
|---|---|
| Import jobs | *Imports* (Also *Importing a file from a storage*) |
| Export jobs | *Exports* |
| Thumbnail jobs | *Thumbnail settings* |
| Shape update/Essence version jobs | *Shapes* |
| File actions | *Files* |
| Sequence rendering | *Item sequences* |
| Item list job | *Listing items in batch* |
| Shape analyze | *Shape analysis* |

### 6.1.2 Concurrency

The number of jobs that execute in parallel is determined by the `concurrentJobs` configuration property.

#### Job pools

Using job pools, it is possible to decide how many jobs of different priorities that can run concurrently. Job pools are configured using the *job pool configuration resource*.

If no pools have been defined then `<concurrentJobs>` controls the number of concurrent jobs. This is the same setting as the `concurrentJobs` configuration property. So by default the job pool configuration will look like:

```
<JobPoolListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <concurrentJobs>3</concurrentJobs>
</JobPoolListDocument>
```

**Warning:** Jobs with priority `IMMEDIATE` are always started, even if the *max concurrent jobs* limit is reached. This could impact system performance. To execute the job with `IMMEDIATE` priority the user must be a super user, that is, have role `_super_access_user`.

**Note:** The max concurrent job setting will only have an effect if it is lower then the size of all pools combined.

#### Priority pools

To start using priority job pools, the job pool definitions must be configured. Priority job pools make it possible to limit the number of concurrent low priority jobs, to make sure that higher priority jobs are able to start even if there are a large number of low priority jobs waiting to be started.

```
PUT /configuration/job-pool
Content-Type: application/xml

<JobPoolListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <pool>
    <priorityThreshold>HIGH</priorityThreshold>
    <size>2</size>
  </pool>
  <pool>
    <priorityThreshold>LOWEST</priorityThreshold>
    <size>3</size>
  </pool>
</JobPoolListDocument>
```

This configuration will allow at most 3 jobs with a priority of LOWEST to MEDIUM to execute at the same time. It will also allow up to 5 concurrent HIGH/HIGHEST priority jobs, as the second pool will contain jobs with a priority of LOWEST *or higher* (the priority threshold is the lower bound and pools have no upper priority bound.)

If there is no job pool with a priority threshold that matches low priority jobs then such jobs will *not* be started. For example, to only let jobs with a priority of MEDIUM or higher to execute:

```
<JobPoolListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <pool>
    <priorityThreshold>MEDIUM</priorityThreshold>
    <size>3</size>
  </pool>
</JobPoolListDocument>
```

### Dedicated pools

New in version 5.4.

To start using a dedicated job pool, the configuration property *dedicatedJobPool* must be set to true. See *create/modify configuration properties* for more information. The job pools must also be configured. If no pools have been defined, the default behaviour will occur where the <concurrentJobs> controls the number of concurrent jobs, regardless if the *dedicatedJobPool* is set to true or not.

Dedicated job pools make it possible to dedicate certain pools to jobs of certain priority. This can be used to make sure that jobs with all priorities are able to start, regardless of how large the number of other priority jobs that are ready to start.

A job pool, with a priority threshold and a size, defines the upper limit of how many jobs of that priority that are allowed to run in that pool. If there are no ready jobs of that priority, those slots will be used for higher priority jobs.

```
PUT /configuration/job-pool
Content-Type: application/xml

<JobPoolListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <pool>
    <priorityThreshold>HIGH</priorityThreshold>
    <size>2</size>
  </pool>
  <pool>
    <priorityThreshold>MEDIUM</priorityThreshold>
    <size>3</size>
  </pool>
  <pool>
    <priorityThreshold>LOWEST</priorityThreshold>
    <size>5</size>
  </pool>
</JobPoolListDocument>
```

This configuration dedicates certain pools for specific priority jobs. The size of priority threshold HIGH allows 2 jobs of priority HIGH to run in that pool. If there are no jobs with priority HIGH ready to be started, jobs with priority HIGHEST will use those 2 slots.

The size of priority threshold MEDIUM allows for 3 jobs of priority MEDIUM to run in that pool, concurrently with 2 jobs with priority HIGH, as defined by the first pool. If there are no jobs with priority MEDIUM ready to be started, jobs with priority HIGHEST will use those 3 slots. If there are no jobs with priority HIGHEST ready to be started, then the 3 MEDIUM slots will be used by jobs with priority HIGH.

The last defined size of priority threshold LOWEST allows for 5 jobs with priority LOWEST to run concurrently of HIGH and MEDIUM jobs. If there are no jobs with priority LOWEST ready to be started, then jobs with priority

HIGHEST or HIGH or MEDIUM or LOW will start, depending if there are jobs of those priorities ready to start, in that specific order.

If there is no defined job pool with a priority threshold that matches lower priority jobs, then such jobs will *not* be started. For example, to only let jobs with a priority of MEDIUM or higher to execute, remove the job pool with priority threshold LOWEST from the example above. Then jobs of priority LOW and LOWEST will never run.

Jobs of priority IMMEDIATE are always started and therefore do not need to be defined by a pool.

### 6.1.3 Job problems

Jobs will enter the state WAITING if a recoverable problem has occurred. Depending on the problem the system might resolve itself or require manual assistance, for example if the system is out of storage space.

A system with no job problems will report:

```
GET /job/problem HTTP/1.1
Content-Type: application/xml

<JobProblemListDocument xmlns="http://xml.vidispine.com/schema/vidispine"/>
```

A system where the transcoder is unreachable for some reason may report:

```
GET /job/problem HTTP/1.1
Content-Type: application/xml

<JobProblemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <problem>
    <id>31532534</id>
    <type>TranscoderOffline</type>
    <job>VX-172716</job>
  </problem>
</JobProblemListDocument>
```

There can be multiple jobs waiting for a problem to be resolved, for example, in case of transcoder or storage problems. For JavaScript problems there will however be one problem per job, as the problem condition is defined by a step specific for each job.

### 6.1.4 Job tasks

The action performed by a task can be implemented either as a method in a Java class or as a JavaScript. Using JavaScript is recommended for all new applications.

```
POST /task-definition/ HTTP/1.1
Content-Type: application/xml

<TaskDefinitionListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <task>
    <description>A custom JavaScript step</description>
    <script><![CDATA[
// This script does nothing but fail the job
job.fatalFail("Testing job failing");
]]></script>
    <step>10000</step>
    <dependency>
      <previous>false</previous>
      <allPrevious>true</allPrevious>
    </dependency>
    <jobType>PLACEHOLDER_IMPORT</jobType>
```

```
    <critical>false</critical>
  </task>
</TaskDefinitionListDocument>
```

#### Defining new tasks

See *JavaScript tasks* on how to create JavaScript tasks.

#### Task dependencies

The execution order is defined by the step numbers and dependencies of the steps. The `dependency` element defines which steps a specific step depend on. There is also the `parallelDependency` element that defines the dependencies that apply if the step is executing as a parallel step.

| | |
|---|---|
| `allPrevious` = true | The step requires all previous step to finish, before it can start. |
| `previous` = true | The step requires the previous step to finish, before it can start |
| `step` = N | The step requires step number N to finish, before it can start |

#### Visualizing tasks

In order to easily see the dependencies between steps for a particular job type, there is functionality to render the job definition as a graph. In order to render the graph, the Graphviz (http://www.graphviz.org/) package is required.

### 6.1.5 Custom job types

It is possible to define *custom job types*. Each custom job type is defined with a name and an integer identifier. Both of these must be unique. Task definitions can then be added to the job type, in the same way as described above.

#### Example

First we create the job type:

```
POST /task-definition/jobtype/MYCOMPANY_CUSTOM_JOB_TYPE?id=25000 HTTP/1.1
```

```
<TaskDefinitionListDocument xmlns="http://xml.vidispine.com/schema/vidispine"/>
```

Then we can add task definitions to our new job type:

```
POST /task-definition/ HTTP/1.1
Content-Type: application/xml

<TaskDefinitionListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <task>
    <description>A custom JavaScript step</description>
    <script><![CDATA[
    logger.log("My custom job is running!");
]]></script>
    <step>100</step>
    <dependency>
      <previous>false</previous>
      <allPrevious>true</allPrevious>
    </dependency>
    <jobType>MY_CUSTOM_JOB_TYPE</jobType>
    <critical>false</critical>
  </task>
</TaskDefinitionListDocument>
```

After this has been done, we can now run the job:

```
POST /job?type=MY_CUSTOM_JOB_TYPE
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-29</jobId>
  <user>admin</user><started>2016-05-17T12:38:22.999Z</started>
  <type>MY_CUSTOM_JOB_TYPE</type>
  <status>READY</status>
  <priority>MEDIUM</priority>
</JobDocument>
```

## 6.2 JavaScript tasks

A JavaScript task is created by including the JavaScript in the task definition document. To evaluate the script Vidispine uses a *JavaScript engine*. A number of global variables are defined for the script to use, see *Common JavaScript functions*.

In addition for task definitions, there is *the job object*.

### 6.2.1 The job object

The `job` object contains methods for reading and writing metadata for the job that is executing, and also for some job control.

`job.`**`getId`**`()`
Gets the id of the job that is executing.

`job.`**`log`**`(`*description*`)`
Logs a message related to the current job step.

**Arguments**

- **`description`** (*`string`*) – The message to log.

`job.`**`getData`**`(`*key*`)`
Gets the data for the given key.

**Arguments**

- **`key`** (*`string`*) – The key to use when getting the data.

`job.`**`setData`**`(`*key*, *value*`)`
Sets the data for the given key.

**Arguments**

- **`key`** (*`string`*) – The key to use when setting the data.
- **`value`** (*`object`*) – The value to insert. Primitive types will be converted to a string. Arrays will be converted into a comma-separated string.

`job.`**`deleteData`**`(`*key*`)`
Removes the given key from the job data.

**Arguments**

- **`key`** (*`string`*) – The key to delete from job data.

`job.`**`fail`**`(`*errorMessage*`)`
Fails the current step, but the step will be retried (up to five times).

**Arguments**

- **errorMessage** (*string*) – The error message, which will be used to set error cause on the job.

job.**fatalFail**(*errorMessage*)
Fails the current step and job.

**Arguments**

- **errorMessage** (*string*) – The error message, which will be used to set error cause on the job.

job.**getUser**()
Gets the user of the job. It replaces the old way of getting the user name using `job.getData("username")`.

job.**getKeys**()
Returns all the keys from the job data.

New in version 4.16.

job.**containsKey**(*key*)
Checks if the job data contains the key.

**Arguments**

- **key** (*string*) – The key to check.

New in version 4.16.

job.**getDataOrDefault**(*key*, *value*)
Returns the value from the job data or the given default value.

**Arguments**

- **key** (*string*) – The key to use when getting the data.
- **value** (*string*) – The value to return if there is no job data set with the given key.

New in version 4.16.

job.**getStepId**()
Returns the current job step id.

New in version 5.0.

job.**wait**(*reason*)
Sets the job in WAITING state.

**Arguments**

- **reason** (*string*) – An explanation of what the job is waiting for.

job.**waitForJobs**(*reason*, *jobIds[]*)
Sets the job in WAITING state until all jobs in jobIds is finished.

New in version 5.4.

**Arguments**

- **reason** (*string*) – An explanation of what the job is waiting for.
- **jobIds** (*string[]*) – An array of job ids that the job is waiting for to finish.

### 6.2.2 Pausing job execution

A JavaScript job step can pause the execution of the job by calling *job.wait()*. This will set the job in the WAITING *job state* and create a job problem of type `JavascriptProblem`.

To determine if the job execution can be resumed, the script is run again every minute with the variable `checkProblem` set to true. If the job should keep waiting, then *job.wait()* should be called again.

#### Waiting for other jobs

New in version 5.4.

If your job step needs to wait for another job to finish before continuing, the script can call *job.waitForJobs()*. This will set the job in the WAITING *job state* and create a job problem of type `WaitingForJobs`. Once all the jobs are finished this job step will be re-executed.

Using *job.waitForJobs()* can make job execution faster as the job problem will be resolved and the job marked as READY once the the dependant jobs have finished.

#### Example

To have the job wait and later run in wait/check mode:

```
if (checkProblem) {
  if (/* condition is fulfilled */ ...) {
    return;
  }
  // Call job.wait() to indicate that the job should wait more
  // See note above
  job.wait("condition still not fulfilled");

} else {
  // run step as normal
  ....

  if (/* condition is not fulfilled */ ...) {
    job.wait("waiting for condition");
    return;
  }

  // continue job execution
  ....
}
```

To have the job wait for other jobs to finish:

```
// Previous step started some jobs and stored as a comma separated list
let jobIds = job.getData('jobIds').split(',');

for (let i=0; i < jobIds.length; i++) {
    const job = api.path('job').path(jobIds[i]).get();
    if (job.status !== 'FINISHED') {
        // If we find a job that is not finished already, wait for all jobs to finish.
        // Note that this will only happen once, during next invocation of the job␣
↪step
        // all the jobs will be finished.
        job.waitForJobs('Waiting for jobs to finish...', jobIds);
    }
}
```

```
// continue job execution
....
```

### 6.2.3 Vidinet job execution

Jobs can be submitted to Vidinet services, for execution or for cost estimates, using the Vidinet functions on the job object.

`job.`**`vidinetJob`**(*type*, *instruction*, *settings*)
Submits a job to Vidinet and sets the job in VIDINET_JOB state.

**Arguments**

- **`type`** (*`string`*) – The type of Vidinet resource to use.
- **`instruction`** (*`string`*) – The job instruction.
- **`settings`** (*`object`*) – A set of key/value pairs related to the job.
  - **item** - The id of the item that the job relates to.
  - **shape** - The id of the shape that the job relates to. Optional. Overrides `tag`.
  - **tag** - The shape of the item that the job relates to. Default is `original`.
  - **resource** - The specific Vidinet resource to submit the job to.

`job.`**`vidinetCost`**(*type*, *instruction*, *settings*)
Request a cost estimate from Vidinet.

**Arguments**

- **`type`** (*`string`*) – The type of Vidinet resource to use.
- **`instruction`** (*`string`*) – The job instruction.
- **`settings`** (*`object`*) – A set of key/value pairs related to the job.
  - **item** - The id of the item that the job relates to.
  - **shape** - The id of the shape that the job relates to. Optional. Overrides `tag`.
  - **tag** - The shape of the item that the job relates to. Default is `original`.
  - **resource** - The specific Vidinet resource to submit the job to.

**Returns** A Future<CostEstimateType> (https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Future.html) that can be used to retrieve the estimate.

#### Example

To submit a job to a Vidinet service that is not natively supported:

```
var item = ...;
var instruction = "...";

job.vidinetJob("TEST", instruction, {
    item: itemId
});
```

In case a cost estimate is wanted before submitting the job to Vidinet:

```
var item = ...;
var instruction = "...";
var settings = {
  item: itemId
};

if ("true".equals(job.getData("estimate"))) {
  var estimate = job.vidinetCost("TEST", instruction, settings);
  var result = estimate.get(); // blocking call

  var cost = result.getService().get(0).getCost();
  job.setData("app_estimated_cost", cost.getAmount());
} else {
  job.vidinetJob("TEST", instruction, settings);
}
```

### 6.2.4 Example: Update item metadata on import

Start by adding a new task to the import job with the script to execute.

**Note:** If using curl, use `--data-binary` instead of `-d` to make sure all new-line characters are kept in the script.

```
POST /task-definition/
Content-Type: application/xml

<TaskDefinitionListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <task>
    <description>Updating item metadata using a JavaScript task</description>
    <script><![CDATA[
...
]]></script>
    <step>10000</step>
    <dependency>
      <previous>false</previous>
      <allPrevious>true</allPrevious>
    </dependency>
    <jobType>PLACEHOLDER_IMPORT</jobType>
    <critical>false</critical>
  </task>
</TaskDefinitionListDocument>
```

```
// Retrieve the id of the item that is being imported
var itemId = job.getData("itemId");
var shapeId = job.getData("originalShapeId");

// Retrieve the shape information
var shape = api.path("item/"+itemId+"/shape/"+shapeId).get();
var video = shape.videoComponent.length;
var audio = shape.audioComponent.length;

// Build a document with the metadata to set
var metadata = {
  "timespan": [
    {
      "start": "-INF",
```

```
      "end": "+INF",
      "field": [
        {
          "name": "title",
          "value": [
            {
              "value": "Item with "+video+" video and "+audio+" audio tracks"
            }
          ]
        }
      ]
    }
  ]
};

// Update the item metadata
var result = api.path("item/"+itemId+"/metadata").input(metadata).put();
var metadata = result.item[0].metadata;
```

### 6.2.5 Example: Update item metadata on import using XML

Scripts can also use ECMAScript for XML (E4X) to easily create and parse XML documents. Using E4X the above script could be written as below. Note that the XML responses from Vidispine will automatically be parsed into E4X XML objects instead of being returned as strings.

```
// Set the default XML namespace so that the Vidispine namespace does not have
// to be specified when retrieving properties or when building the metadata document
default xml namespace = "http://xml.vidispine.com/schema/vidispine";

// Retrieve the id of the item that is being imported
var itemId = job.getData("itemId");
var shapeId = job.getData("originalShapeId");

// Retrieve the shape information
var shape = api.path("item/"+itemId+"/shape/"+shapeId).dataType("xml").get();
var video = shape.videoComponent.length();
var audio = shape.audioComponent.length();

// Build a document with the metadata to set
var metadata = <MetadataDocument>
  <timespan start="-INF" end="+INF">
    <field>
      <name>title</name>
      <value>Item with {video} video and {audio} audio tracks</value>
    </field>
  </timespan>
</MetadataDocument>

// Update the item metadata
var result = api.path("item/"+itemId+"/metadata").input(metadata).put();
var metadata = result.item[0].metadata;
```

## 6.3 Task groups

Task groups can be used to control the transcoders that a specific job should use, or to control the number of concurrent jobs running by job type. It may be expanded in the future to include not only jobs and transcoders, but also other

types of tasks and resources.

- A task group identifies a set of jobs and the resources available to those jobs.
- Jobs are identified by a criteria on the group.
- A job can belong to multiple groups, but only a single group for each type of resource. If a job satisfies the criteria on multiple groups, then the job belongs to the group with the highest priority.
- A transcoder can belong to any number of groups.
- A job will only use resources from the group(s) that it belongs to.

### 6.3.1 Creating a task group

Task groups are referred to by name. Each group should specify a job criteria and a number of transcoders, and a priority if needed.

```
PUT /task-group/imports
Content-Type: application/xml

<TaskGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <job>
    <type>PLACEHOLDER_IMPORT</type>
  </job>
  <transcoder>
    <id>VX-1</id>
  </transcoder>
  <transcoder>
    <id>VX-2</id>
  </transcoder>
  <priority>MEDIUM</priority>
</TaskGroupDocument>
```

### 6.3.2 Task group criteria

Task groups can have multiple criteria. A job must then satisfy them all to be considered being part of that group. The selections in a criteria form a logical OR.

For example, to restrict jobs that are either imports or exports, and from the admin or bulk-import user, to transcoder VX-1:

```
<TaskGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <job>
    <type>PLACEHOLDER_IMPORT</type>
    <type>EXPORT</type>
  </job>
  <job>
    <user>admin</user>
    <user>bulk-import</user>
  </job>
  <transcoder>
    <id>VX-1</id>
  </transcoder>
</TaskGroupDocument>
```

This is evaluated as:

```
(type:PLACEHOLDER_IMPORT OR type:EXPORT)
AND
(user:admin OR user:bulk-import)
```

#### Job criteria

Jobs can be matched on:

- Priority - To include jobs with a certain priority.
- Type - To include jobs of a certain type.
- User - To include jobs created by a specific user.
- Group - To include jobs created by a user in a specific group.
- Data - To include jobs with certain data.

### 6.3.3 Task group priority

A job will use resources from the task group with the highest priority and a matching criteria. If two task groups have the same priority then the groups are ordered by name in alphabetical order, and the first one is picked.

### 6.3.4 Task group concurrency limit

New in version 5.2.2.

By setting the `maxConcurrency` setting in a task group it's possible to control the maximum number of concurrent jobs for any matching job. If a job matches multiple task groups, the one with the highest priority and the lowest `maxConcurrency` value will be effective.

For example, to specifically limit the amount of concurrent jobs of type `EXPORT`, the following task group can be created:

```
PUT /task-group/export-job-limit
Content-Type: application/xml

<TaskGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <job>
    <type>EXPORT</type>
  </job>
  <priority>MEDIUM</priority>
  <maxConcurrency>3</maxConcurrency>
</TaskGroupDocument>
```

Other means to control the maximum number of concurrent jobs includes using *job pools* and the *concurrentJobs* configuration property.

### 6.3.5 Job problems

If a job cannot run because the transcoders available to it are offline, then a transcoder offline problem will be created. The problem will contain the name of the group and the job id(s).

This allows you to see which group/transcoder(s) a job is blocked on. For example:

```
GET /job/problem
```

```
<JobProblemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <problem>
    <id>35113</id>
    <type>TranscoderOffline</type>
    <job>VX-115770</job>
    <data>
      <key>taskGroup</key>
      <value>imports</value>
    </data>
  </problem>
</JobProblemListDocument>
```

CHAPTER

SEVEN

# NOTIFICATIONS

Notifications are sent from the system when predefined events occur. An example of such an event could be a job that finishes. Examples of when this could be useful are:

- Getting a notification when a job finishes.
- Making sure the metadata input for a certain field is correct.

Notifications involve a quadruple:

1. The resource or entity to be notified about.
2. The event that should trigger the notification.
3. The action that should be taken when the notification is triggered.
4. Filters that further specifies the behavior of the trigger.

## 7.1 Resources

A number of different entity types support notifications. Below is a short description of the different entity types and what events can trigger a notification:

- **Items** – notifications can trigger on item delete/create, metadata changes, shape changes and access control changes.
- **Collections** – can trigger on creation/deletion, metadata changes and content changes.
- **Jobs** – can trigger on job create, update, finish, fail and stop.
- **Groups** – group notifications can trigger on group create, delete and modify.
- **Storages** – can trigger on storage create/delete, and on new files.
- **Files** – can trigger on group create, delete and modify.
- **Quota** – quota notifications can trigger on quota create, delete, and quota exceeded warnings.
- **Document** – can trigger on document create, delete and modify.

## 7.2 Actions

An action is what will be done when a notification is triggered. The action can either be to:

- Perform a HTTP request.
- Invoke a Java class method.
- Send a JMS message.

- Execute a JavaScript.
- Send a message to Amazon SQS.
- Send a message to Amazon SNS.

The data included in the request or message will be multivalued key-value data identifying the event that has occurred.

An action can be sent either synchronous or asynchronous. In the case of a synchronous action the message will be sent in the same thread as where the notification is triggered. And execution will only continue if the recipient acknowledges and approves the message. In the asynchronous case the message will be sent in another thread and execution will continue immediately.

For a full description of actions, refer to the API reference on *Actions*.

## 7.3 Triggers

A trigger is the event that will cause the notification to perform its action. Different triggers exist for different resources. The trigger used determines what output that can be expected. Below an overview of available triggers can be seen:

- Item triggers.
  - Shapes
  - Metadata
  - ACLs
- Collection triggers
- Group triggers
- Job triggers
- Storage triggers
- File triggers
- Quota triggers
- Document triggers.
- Deletion lock triggers.
- Placeholder (null) triggers.

The placeholder trigger is simply the lack of a trigger-type. For a full description of triggers, refer to the API reference on *Triggers*.

## 7.4 Job filtering

### 7.4.1 Job types

Filter criteria can be added to job notifications in order to filter which type of jobs they trigger on.

### 7.4.2 Job metadata

Either by string comparison or regular expressions.

### 7.4.3 Filters

Filters can be used to specify the trigger further. For example in the case of metadata, the notification can be filtered to only trigger for certain values.

CHAPTER

EIGHT

# RESOURCES

Resources in Vidispine are components used for auxiliary storage or transformation. The two most commonly used resource type are the `thumbnail`, which is used to store thumbnails, and `transcoder`, which points to instances of the Vidispine transcoder.

## 8.1 Transcoders

When you import items the Vidispine transcoder will be used to detect the type of media that is being imported and, of course, to transcode the media to any formats that you have requested.

The common operations performed by the transcoder are:

- Media shape deduction
- Transcoding
- Sequence rendering
- Partial file extraction
- XMP extraction and rewrite

The Vidispine transcoder has a REST API that Vidispine uses to perform the above operations. This API is not described in this document, as it typically should not be accessed directly.

### 8.1.1 Adding a transcoder

Add a transcoder by creating a new `transcoder` resource. The resource document should contain information on how to reach the transcoder and what storages the transcoder has direct access to.

```
POST /resource/
Content-Type: application/xml

<?xml version="1.0"?>
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <transcoder>
    <url>http://transcoder.example.com:8888/</url>
    <directAccess>
      <filter>file:/srv/media/.*</filter>
    </directAccess>
  </transcoder>
</ResourceDocument>
```

Vidispine checks the status of transcoders continuously in the background. As such, if the configuration is correct you will see that the transcoder shows up as online in a few seconds.

```
GET /resource/VX-7
```

```
<?xml version="1.0"?>
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-7</id>
  <transcoder>
    <url>http://transcoder.example.com:8888/</url>
    <directAccess>
      <filter>file:/srv/media/.*</filter>
    </directAccess>
    <state>ONLINE</state>
  </transcoder>
</ResourceDocument>
```

The Vidispine installer will by default install and configure a transcoder in Vidispine for you, so this step is typically not needed.

### 8.1.2 Using multiple transcoders

Depending on your license, you may be allowed to use more than one transcoder. To do so, simply add additional transcoders as explained above. Vidispine will submit transcode jobs to the transcoder based on the current number of jobs being processed by the transcoder.

Vidispine will use the transcoder with the least amount of work. If a transcoder goes offline then any transcode job steps using that transcoder will fail and be retried using one of the online transcoders. If all transcoders are offline then jobs will *wait* for one to become available.

---

**Note:** The *clusterName* property must be set if multiple Vidispine installations are to share a transcoder. Each installation must have a unique cluster name. This applies regardless if the installations have the same site name or not.

---

### 8.1.3 How transcoders perform jobs

A transcoder will perform a job as soon as it is received, and will not schedule jobs for later execution. Vidispine, that is, the user of the transcoder is responsible for scheduling which jobs a transcoder should execute and when they should be executed.

### 8.1.4 Transcoder job limit

The `maxJob` setting can be used to limit the number of Vidispine jobs that may use a specific transcoder at the same time. If all transcoders are busy, jobs will be put on `WAITING` state, with a `TranscoderBusy` problem. The jobs will restart as soon as any qualified transcoder becomes available again.

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-7</id>
  <transcoder>
    <url>http://transcoder.example.com:8888/</url>
    <maxJob>5<maxJob>
  </transcoder>
</ResourceDocument>
```

A Vidispine job typically triggers multiple transcoder jobs, e.g. shape deductions and transcodes, so in the above case there may still be more than 5 running jobs on a transcoder. They will however all belong to at most 5 jobs in Vidispine.

The above setting works for VSA transcoders as well. VSA users can also add `transcoder.maxJob` to one of the agent configuration files. For example:

```
$ cat /etc/vidispine/agent.conf.d/transcoder.conf

transcoder.maxJob=5
```

### 8.1.5 The transcoder's configuration file

The transcoder configuration file `config.xml` contains default settings for the transcoder and need typically not be modified, as the settings can instead by configured in Vidispine.

#### Modify the transcoder file

On a Linux system, copy the file `/opt/vidispine/transcoder/config.xml` to `/etc/transcoder-config.xml`. Then edit `/etc/transcoder-config.xml`. The file in `/etc` takes precedence over the file in `/opt/vidispine`.

#### Modify the transcoder resource

On all operating systems, the transcoder configuration can be changed by adding configuration to the resource definition of the transcoder (*Adding a transcoder*).

Note that port of the transcoder cannot be changed in this fashion.

Modifying the transcoder configuration is this fashion takes precedence over the local configuration file and the global transcoder configuration, see below.

#### Modify all transcoders

It is also possible to change the configuration of all transcoders, by setting the configuration property `transcoderDefaultConfiguration` to the XML representation of the transcoder configuration.

#### Thumbnail settings

---

**Note:** The preferred way of changing the thumbnail and poster settings is by changing the appropriate values in the TranscodePresetDocument in a *shape tag*. For example, by changing the `thumbnailResolution` and `thumbnailPeriod` elements. The setting in shape tag have priority over the transcoder setting.

---

The `thumbnailResolution` element contains the default resolution of the thumbnails produced by the transcoder.

```
<a:thumbnailResolution>
   <a:width>320</a:width>
   <a:height>240</a:height>
</a:thumbnailResolution>
```

You can also change the thumbnailing frequency by changing `thumbnailPeriod`. For example, to thumbnail every 3 seconds:

```
<a:thumbnailPeriod>
   <a:samples>3</a:samples>
   <a:timeBase>
      <a:numerator>1</a:numerator>
      <a:denominator>1</a:denominator>
   </a:timeBase>
</a:thumbnailPeriod>
```

If the transcoder does not use a scene change detection plugin, the frequency defaults to once every 10 seconds.

### StatsD settings

To have the transcoder send metrics to a StatsD server you can either:

- Enable StatsD using the API, see *StatsD*
- Update the transcoder configuration with the address and port of the StatsD server:

```
<a:statsd>
  <a:destination>
    <a:address>127.0.0.1</a:address>
    <a:port>8125</a:port>
  </a:destination>
  <a:prefix>t1</a:prefix>
</a:statsd>
```

The `prefix` element configures the prefix to use for each metric. By default this is the `transcoder`.

### Transcoder resources settings

### Path to temporary storage

Controls where temporary files are stored. Default is `/tmp` on UNIX-like systems, or `%TEMP%` on Windows.

```
<a:tempPath>/mnt/largetemparea<a:tempPath>
```

### Number of decoding threads

Controls the number of decoding threads. Defaults to `4` for I-frame-only formats. The actual number of threads used depends on codec.

```
<a:decoderOfferThreads>8<a:decoderOfferThreads>
```

### Number of encoding threads

Controls the number of encoding threads. Defaults to automatic setting. The actual number of threads used depends on codec.

```
<a:encoderThreads>8<a:encoderThreads>
```

### HTTP buffer sizes

Controls the size of HTTP reads and writes of the transcoder. `dataBufferSize` controls the maximum number of read bytes in memory. Default is 100 MB. `dataBufferWriteSize` controls the maximum number of write bytes in memory. Default is 100 MB. `dataBufferFlushTime` controls the number of seconds written bytes are stored in memory before it is flushed. Default is 4 seconds.

For system that uses segment files and where the transcoder has enough memory, it is recommended to increase these numbers, up to 10 times.

### Image processing

To control the memory and disk usage used by the transcoder for image processing, use the `<imagemagick>` element in the transcoder configuration. The most important settings are listed below, for a complete list, see

http://www.imagemagick.org/script/resources.php (under environment variables, used without the MAGICK_ prefix in the transcoder configuration).

#### Maximum heap usage

```
<a:imagemagick>
  <a:key>MEMORY_LIMIT</a:key>
  <a:value>1GB</a:value>
</a:imagemagick>
```

#### Temporary work area

```
<a:imagemagick>
  <a:key>TEMPORARY_PATH</a:key>
  <a:value>/var/tmp</a:value>
</a:imagemagick>
```

The default value is the value set by the general transcoder temporary path, see above. It is recommended that the tempPath setting is used, rather than the imagemagick one.

### 8.1.6 Operations overview

#### Zeroconf transcoders

The following is done to remove the need to configure the transcoders directly:

- Vidispine pushes its own license to the transcoder, so that each transcoder does not need a license file of their own.
- The transcoder returns the IP address from where the license was pushed, that is, the IP address of the application server, removing the need for explicitly configuring the reverse address, that is, where the transcoder can reach Vidispine, in most cases.
- In addition, Vidispine generates temporary pre-authorized URIs that are used by the transcoder. This removes the need for entering any application server information in the transcoder configuration file.

#### Reverse address and NAT

The reverse address does not work if there is NAT or other port forwarding mechanisms between the application server and the transcoder. If so, the address to VS-EA can be overridden in the definition for the transcoder by setting the <reverseAddress> element.

```
<?xml version="1.0"?>
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <transcoder>
    <url>http://transcoder.example.com:8888/</url>
    <reverseAddress>vs.example.com</reverseAddress>
  </transcoder>
</ResourceDocument>
```

The rules for how the address to Vidispine is determined are as follows:

1. If the configuration property *apiNoauthUri* is set, it is used for all transcoders.
2. If the configuration property apiNoauthPort is set, it is used for together with the detected or manually set reverse address.

#### Transcoder's access to media

By default, the transcoder accesses non-file-schema media through the application server. This has several advantages:

- The same user is used for all file access.
- Possibility for support for extended file attributes and permissions.
- Support for other file systems (URI schemes).

Streaming the media puts some extra load on the application server. Some tuning might be necessary.

The transcoder resource in Vidispine can be set up to access files directly. By adding a `directAccess` element to the transcoder resource, Vidispine will let the transcoder access the media directly. If no `directAccess` elements are present, an implicit

```
<directAccess>
   <filter>file:.*</filter>
</directAccess>
```

is added. In order to tell Vidispine that all files should go via the application server, add an

```
<directAccess>
   <filter>NO_MATCH</filter> <!-- dummy regular expression that does not match␣
↪anything -->
</directAccess>
```

#### Growing files

For all file systems that supports read-while-write, and for container formats that are built for streaming (e.g. MXF), growing file is supported when streamed through the application server. If growing files is required to local files with the `file` scheme, a `directAccess`/`NO_MATCH` element as per above must be added to the resource configuration.

## 8.2 Transcoder discovery

Vidispine can automatically discover transcoders using either HTTP or DNS. This makes it possible to use Consul (http://consul.io/), Amazon Route 53 (http://aws.amazon.com/route53/) or any DNS or HTTP server to track the available transcoders, with custom health checks and rules to determine which transcoders should be used by Vidispine for example.

You could also configure Vidispine instances to read transcoders from another instance, as an easy way to manage a set of transcoders.

### 8.2.1 Adding a transcoder directory

To have Vidispine discover transcoders, add a transcoder resource to Vidispine with the type set to `DIRECTORY`.

```
POST /resource/
Content-Type: application/xml

<?xml version="1.0"?>
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <transcoder>
    <type>DIRECTORY</type>
    <url>http://service-user:oeHie2Ye@vs1.example.com:8080/API/resource/transcoder</
↪url>
  </transcoder>
</ResourceDocument>
```

Once the transcoders have been retrieved, they will show up as nested transcoders under the transcoder resource.

```
GET /resource/VX-24
```

```
<?xml version="1.0"?>
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-24</id>
  <transcoder>
    <type>DIRECTORY</type>
    <url>http://service-user:oeHie2Ye@vs1.example.com:8080/API/resource/transcoder</
↪url>
    <state>ONLINE</state>
    <transcoder>
      <url>http://t1.transcoder.example.com:8888/</url>
      <version>4.4</version>
      <reverseAddressDetected>172.17.42.1</reverseAddressDetected>
      <state>ONLINE</state>
    </transcoder>
    <transcoder>
      <url>http://t2.transcoder.example.com:8888/</url>
      <state>OFFLINE</state>
    </transcoder>
  </transcoder>
</ResourceDocument>
```

**Note:** If the HTTP/DNS server is offline then the known set of transcoders will be kept and used until the server comes online again and the set of transcoders is updated.

### 8.2.2 Supported URIs

**http:**

**Syntax** `http://[{user}:{password}@]{host}/{path}`

**Response** `application/xml` - ResourceListDocument

The HTTP server should return a list with all available transcoders.

**dns:**

**Syntax** `dns:[//{dnsServer}/]{domainName}`

Vidispine will perform a SRV lookup to retrieve the host and port of all available transcoders. SRV lookups will be done against:

- `{domainName}`
- `_transcoder._http.{domainName}`
- `_transcoder._https.{domainName}`

#### RFC 2782 names

If the domain name already has the format of a RFC 2782 style SRV resource record (`_service._protocol.name`), then a single SRV lookup will be done.

If the service or protocol name is `https` then HTTPS will be used instead of HTTP to connect to the discovered transcoders. For example, these SRV records would enable secure communication using HTTPS between Vidispine and transcoders:

- `_transcoder._https.example.com`
- `_https._tcp.transcoder.example.com`

Changed in version 4.15: Support for RFC 2782 style lookups and HTTPS was added.

## 8.3 External transcoders

Using the external transcoder support in Vidispine it is possible to use transcoders from other companies, or to perform transcodes in other ways. This is done using watch folders.

- With transcoders that support watch folders directly, it's simply a matter of configuring both Vidispine and the external transcoder to use the same watch folder.
- Transcoders that do *not* support watch folders can still be integrated with by writing a service that monitors the watch folder and sends transcode request to the external transcoder accordingly.

---

**Important:**

- It is not possible to transcode using the Vidispine transcoder and an external transcoder at the same time.
- It is only possible to transcode using one external transcoder shape tag at the time.

---

Any method supported by Vidispine can be specified as the source or destination, meaning that the watch folders do not need to be local.

### 8.3.1 How it works

When starting an import or transcode job, Vidispine will check if the given shape tag is defined to be handled by an external transcoder. If it is, then the source file (e.g. the original essence of the item) will be copied to the transcoder's watch folder (e.g. `<source>` the external-transcoder *ResourceDocument*); then the job waits for one or more files to appear in the destination folder (e.g. `<destination>` in *ResourceDocument*), and perform the rest steps. Note: only the transcode step is handled by the external transcoder.

#### Settings

**Filename pattern** It is mandatory to define a filename pattern (a.k.a `<regex>`) in the external transcoder resource to control what files the job should look for. In order to support multiple transcodes at the same time, the regex will be prefixed using the file name of the essence automatically. That is:

If the original essence file name is `VX-100`, and the regex is `.*output.*`, then Vidispine will look for files matching `\QVX-100\E.*output.*`.

**Timeout** The output file must appear in the destination folder within this timeout, or the transcode step will be marked as failed. The default timeout is 30 seconds.

**Interval** How frequently the destination folder should be checked for new or updated files. The default interval is 5 seconds.

**Checks** How many times an output file must remain unchanged for the file to be considered completely written. By default files must remain unchanged for 3 checks.

### 8.3.2 Adding an external transcoder

Add an external transcoder by creating an `externalTranscoder` resource using *`POST /resource`*.

```
POST /resource/externalTranscoder/
Content-Type: application/xml

<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <externalTranscoder>
        <source>file:///mnt/external-transcoder/source/</source>
        <destination>file:///mnt/external-transcoder/destination/</destination>
        <shapeTag>external-format</shapeTag>
        <timeout>60000</timeout>
        <regex>.*demo.*</regex>  <!-- Since Vidispine 4.0 -->
   </externalTranscoder>
</ResourceDocument>
```

### 8.3.3 Using an external transcoder

Before starting a transcode, make sure the shape tag in the example, has been defined in an external transcoder resource.

```
POST /shape-tag/external-format
Content-Type: application/xml

<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mp4</format>
  <audio></audio>
  <video></video>
</TranscodePresetDocument>
```

The external transcoder is supported in `AUTO_IMPORT` and at the following requests

- *POST /import*
- *POST /import/raw*
- *POST /item/(item-id)/transcode*
- *POST /item/(item-id)/shape/(shape-id)/transcode*

## 8.4 Thumbnail resources

A thumbnail resource defines a location where the thumbnails will be stored. For details how the thumbnails are stored and for the supported location types see *How thumbnails are saved on disk*.

### 8.4.1 Adding a thumbnail resource

Add a thumbnail resource using *POST /resource*.

```
POST /resource
Content-Type: application/xml

<?xml version="1.0"?>
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <thumbnail>
    <path>file:///srv/thumbnails/</path>
  </thumbnail>
</ResourceDocument>
```

### 8.4.2 Reading thumbnails

The thumbnails in that directory will then be available from the API as described on *Thumbnail resource handling*. For example, all thumbnails can be listed using `GET /thumbnail/(resource-id)`.

```
GET /thumbnail/VX-2
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>VX-1</uri>
  <uri>VX-3</uri>
  <uri>VX-4</uri>
  <uri>VX-7</uri>
</URIListDocument>
```

However, you would typically not access thumbnails from that resource directly. Instead, fetch thumbnails for an item using *`GET /item/(item-id)/thumbnailresource`* or using the `thumbnail` content parameter.

```
GET /item/VX-7/thumbnailresource
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>http://localhost:8080/API/thumbnail/VX-1/VX-7;version=0</uri>
</URIListDocument>
```

```
GET http://localhost:8080/API/thumbnail/VX-1/VX-7;version=0
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>0@PAL</uri>
</URIListDocument>
```

### 8.4.3 Thumbnail resource permissions

A thumbnail resource can be made read-only or read-protected, for example:

- When migrating thumbnails from one location to another; to have new thumbnails written to one resource but old thumbnails still read from another resource.
- As read-protected, in case the disk where thumbnails are stored needs to be taken offline. VS will then avoid serving thumbnails from that resource.

This is done using the `mode` element.

```
<?xml version="1.0"?>
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <thumbnail>
    <path>file:///srv/thumbnails/</path>
    <mode>NONE</mode>
  </thumbnail>
</ResourceDocument>
```

Allowed values are:

- `READ_WRITE` - full access (default)
- `READ` - read-only
- `NONE` - no access.

### 8.4.4 How thumbnails are saved on disk

The thumbnails can be stored either in a database form or as one file per thumbnails.

Thumbnails are stored in the resolution and format as requested when the thumbnails were created, and it's not possible to for example request a thumbnail as a PNG if it has previously been created as a JPEG.

#### Database

The thumbnail path as specified in the ResourceDocument should have the format

- path (e.g. `/srv/media/thumbnails/`), or
- file URI (e.g. `file:///src/media/thumbnails/`)

Thumbnails are stored in a separate directory and database - one for each item. Vidispine will automatically migrate the databases during runtime if necessary, so no special action is required when updating Vidispine to a newer version or when restoring an old thumbnail backup on a newer system.

#### One file per thumbnail

The thumbnail path as specified in the ResourceDocument should have the format

- URI with the `direct+` prefix (e.g. `direct+file:///src/media/thumbnails/`

All URIs supported as *Storage method URIs* are supported.

#### Using a tree structure for thumbnails

Putting all files in the same directory of a storage can cause degraded performance on some file systems.

By setting the configuration property *`thumbnailHierarchy`*, the naming convention for the thumbnails' folders is changed to **site-id** – **number1** / **number2**. The number set in *`thumbnailHierarchy`* controls the size of **number2**.

The *`thumbnailHierarchy`* works in the same way as *`fileHierarchy`* does for files. See *Using a tree structure for files* for an example. The property works both for the database thumbnail storage and the direct thumbnail storage.

> **Warning:** Changing the *`thumbnailHierarchy`* property **will cause old thumbnails to be lost**. If you need to change the value on a system in production, please contact Vidispine.

## 8.5 Vidispine Server Agent

The Vidispine Server Agent, *VSA*, is a daemon process running on servers connecting to a Vidispine Server, *VS*. VSA is composed of a *Vidispine Transcoder* and the VSA supervisor.

### 8.5.1 How to install VSA

#### Prerequisites

- A running VS instance, version 4.4 or newer
- A server running Ubuntu 14.04 or higher, 64-bit, or CentOS 6.5 or higher, 64-bit

#### Installation

Add the Vidispine repository according to the documentation on repository (http://repo.vidispine.com/). Then you can install and start VSA. With Ubuntu/Debian:

```
$ sudo apt-get install vidispine-agent vidispine-agent-tools
```

With CentOS/RedHat:

```
$ sudo yum install vidispine-agent vidispine-agent-tools
```

After that, the agent can be connected to Vidispine server.

### 8.5.2 Connecting to Vidispine

The agent can then be connected either with or without establishing an SSH tunnel to Vidispine server. The latter should be used if an encrypted network connection has already been established to Vidispine server, or if the server and the agent runs within the same network.

- *Connecting with SSH tunnel*
- *Connecting without SSH tunnel*

#### Connecting with SSH tunnel

The configuration files are located in `/etc/vidispine/`. Configuration can be stored in either the file `agent.conf` in this directory, or in files in the subdirectory `agent.conf.d`. It is recommended that a file is created in the `agent.conf.d` directory. Specifically, there are two setting that has to be set: the connection to VS, and the unique name of the VSA server. The first one you will get from the Vidispine instance.

1. Enable the Vidispine VSA port, by adding this to the `server.yaml` file (change the port number as necessary). The server will need to restart for any changes to take effect.

```
vsaconnection:
   bindPort: 8183
```

**Note:** This step is new in Vidispine 4.6.

2. On the Vidispine instance, install the `vidispine-tools` package and run

```
$ sudo vidispine-admin vsa-add-node
```

**Note:** In Vidispine 4.6, the command has changed to `vsa-add-node`. With the new vsa-add-node command, one VSA can connect to multiple vidispine-servers.

3. Fill in the user name, password and IP address. Enter the unique name, but you can leave the UUID empty.
4. Now, on the VSA server, add this information to `/etc/vidispine/agent.conf.d/connection`.
5. Start VSA:

```
$ sudo service vidispine-agent start
$ sudo service transcoder start
```

6. Wait 30 seconds. Now verify that it is connected:

```
$ sudo vidispine-agent-admin status
```

Agent, transcoder and Vidispine should all be ONLINE.

#### Connecting without SSH tunnel

1. Create a file `/etc/vidispine/agent.conf.d/custom.conf` with content like:

```
userName=admin
password=KioqYWRtaW4=
directVSURI=http://172.17.0.7:8080/
vsaURI=http://172.17.0.8:8090/
```

   - `userName`: Vidispine user name.
   - `password`: Base64 encoded value of a `***` prefixed password. For example, the value should be the result of `echo -n ***admin | base64`, if the password is `admin`.
   - `directVSURI`: the address VSA uses to connect to Vidispine server.
   - `vsaURI`: the address that can be used by Vidispine server to connect to VSA

2. Restart VSA:

```
$ sudo service vidispine-agent restart
```

3. Wait 30 seconds. Now verify that it is connected:

```
$ sudo vidispine-agent-admin status
```

   Agent, transcoder and Vidispine should all be ONLINE.

4. Also, the VSA should listed under the server:

```
$ curl -X GET -uadmin:admin http://localhost:8080/API/vxa
```

### 8.5.3 Adding a share

On the VSA, run the following command:

```
$ sudo vidispine-agent-admin add-local-share
```

This will add a share in VSA, and create a storage in VS. You can verify this by listing the storages (*List all storages*). The storage is listed with a method that has a `vxa:` URI scheme. The UUID (server part) of the URI matches the UUID from `vidispine-agent-admin` status.

**Warning:** If the share is removed from the VSA, the storage will be automatically deleted from VS, including all file information (but not the files themselves). In order to keep the storage, e.g., if the storage is moved from one VSA to another, remove the `vxaId` metadata field from the storage.

#### Enable write access

When a new share is added, the storage method is marked as read-only. To enable writing to the share:

- set the write field of the method to true, and
- change the storage type to LOCAL (meaning it can be a target for all file operations)

#### Associate many VSAs to one storage

It is possible to have several VSA nodes serving one shared file system. This can be used for increasing transcoding capability or to generated redundancy.

1. Add the share individually on all VSAs (see above). This will generate as many storages as there are VSAs.
2. Now copy the storage methods from all but the first storage to the first storage.
3. On the first storage, remove the `vxaId` storage metadata (see above).
4. Remove all but the first storage.

### 8.5.4 VSA and S3 credentials

A VSA transcoder can be given *direct access* to S3 storages, meaning the agent will access the files directly without them being proxied by the main server. If the configuration property *useS3Proxy* is set to `true`, pre-signed URLs will be used for agents to read S3 objects. If it is set to `false`, or if it is a `WRITE` operation, AWS credentials will be sent to agents.

The type of AWS credentials being sent to the agents can be controlled by the configuration property *s3CredentialType*:

- `secretkey`: The access key and the secret access key configured in the S3 storage URI will be sent to the agent.
- `temporary`: The AWS Security Token Service (STS) will be used to generate temporary credentials to send to the agents. The duration of the credentials is controlled by *stsCredentialDuration*. You can set *stsRegion* to control in which region Vidispine server will call the AWS Security Token Service (STS) API.
- `none`: No credentials will be sent to the agent. The agent then needs to rely on a local `AwsCredentials.properties` file, or an IAM role on the instance to access S3 objects.

There is also a configuration entry called `s3CredentialType` available in the `agent.conf`, that can be used to configure this behavior on a per-agent basis.

The final *effective* credential type will be the **min** of *Server s3CredentialType* and *Agent s3CredentialType*. And the order of the values is `secretkey > temporary > none`.

For example, no credentials will be sent to the agent, if an agent has the following configuration:

```
s3CredentialType=none
```

and the server has:

```
<property lastChange="2014-07-14T14:55:15.432+02:00">
  <key>s3CredentialType</key>
  <value>temporary</value>
</property>
```

**Note:** For an older agent to work with 4.14 server, the credential type on the server side has to be set to either `secretkey` or `none`.

### 8.5.5 Agent properties

Configuration properties that can be used in the agent configuration file. Upon start, configuration is read from `/etc/vidispine/agent.conf` and any files in the directory `/etc/vidispine/agent.conf.d`.

### Basic

**vxaName** The name the VSA is using to register itself. Optional but recommended. With a name set, the name can be used instead of UUID in `vxa://` URIs.

**operationMode** Should always be `VSA-VS`.

**uuid** The UUID of the VSA. Must be unique and follow the UUID syntax.

### Connection

**agentGroup** String that is used to signal to Vidispine server that all agents in the same group can reach each other.

**bindAddressV4/bindAddressV6** The network address that the agent should accept connections on. If not set, `127.0.0.1` is used.

**vxaPort** The network port that the agent should listen on. Default 8090.

**externalUri** URI that the agent can be reached at. For example: `http://10.0.0.20:8090/`, `https://vsa.example.com/`.

**connectionString, connectionString1, connectionString2...** How the VSA should connect VidiCore. Generated by `vidispine-agent-admin`.

**directVSURI, directVSURI1, directVSURI2** If VSA can connect directly to VidiCore (without secure tunnel), this is the URI to VidiCore (from VSA).

**vsaURI** If VidiCore can connect directly to VSA (without secure tunnel), this is the URI to VSA (from VidiCore).

**userName** User name used to connect to VidiCore. Not recommended. Use `vidispine-agent-admin` to create a secure connection instead.

**password** Password used to connect to VidiCore. Not recommended. Use `vidispine-agent-admin` to create a secure connection instead.

**sshProxy** Proxy (http, socks4, socks5) used for SSH connection. Not required for new connections created by `vidispine-agent-admin`.

**fingerPrint** SSH fingerprint of SSH server on VidiCore side. Connection will fail if fingerPrint is set and no matching. Default is that connection is allowed, but a warning is emitted in the log file.

**pingInterval** How often the VSA should contact VidiCore. Default is `4` seconds, but can be increased to lower traffic. Recommended: `60`.

**restSelectorRunners** The number of threads that will be available to serve incoming requests. The selector runner will delegate the actual work that should be done to a worker thread.

New in version 5.3.

**restWorkerThreads** The number of worker threads that are available. These threads carry out the actual work in the VSA. For example they handle transfer jobs performed by the VSA. They typically also deliver results of requests sent to the VSA. However, see also transfer section below.

New in version 5.3.

### Logging

**logLevel** Overall log level. Accepted values are `ALL`, `TRACE`, `DEBUG`, `INFO` (default), `WARN`, `ERROR`, `FATAL`, `OFF`.

**logLevel.(class or package)** Class or package-specific logging.

#### Transfer jobs

**transferThreadCount** Use multiple threads for a single transfer. Can speed up S3 transfers significantly. Default is 1 (single thread).

New in version 5.4.

**transferBufferSize** Size of transfer chunk used in transfer jobs. Default is 10000000 (10 MB).

New in version 5.4.

**checkTransferDestination** If set, VSA will wait up to given number of seconds to appear in file listings before reporting the transfer as complete.

**readTransferDestination** If set to true (which is the default), VSA verify a transfer by reading the first byte of the destination before reporting the transfer as complete.

#### Hash compute jobs

**hashThreadCount** Use multiple threads for reading a file during hash computation. The actual computation is still done in one thread. Default is 1 (single thread).

New in version 5.4.

#### Transcoder jobs

**transcoder.maxJob** The sets the maximum transcoder jobs the VSA will process. This is done by setting the maxJob element of the transcoder resource in VidiCore.

**transcoder.directAccess** Controls if the VSA can access the input files directly. Note that there are two level of media access proxying for transcode jobs. VidiCore will proxy all access for the VSA which does not fit the directAccess filter, if the directAccess is set. VSA will proxy media access for the VidiCoder for URIs that are not http or file.

**transcoder.port** How the VSA reaches the transcoder. Should be `8888` unless the transcoder listens to another port.

#### Storage access

**s3...** All S3 configurations listed in *Storage and file* are available as VSA configuration.

**ftppool.maxtotal** Maximum number of entries in FTP connection pool. Default is -1 (unlimited).

**ftppool.maxtotalperkey** Maximum number of entries in FTP connection pool per key (scheme/host/port). Default is -1 (unlimited).

**ftppool.minidleperkey** Keep at least this number of connections idle. Default is 0.

**ftppool.timebetweenevictionrunsmillis** Time between when idle connections are checked for closing, in milliseconds. Default is 30000 (30 seconds).

**ftppool.minevictableidletimemillis** The minimum time an connection is idle before it can be closed, in milliseconds. Default is 60000 (60 seconds).

### 8.5.6 Direct transfers between VSAs

New in version 5.0.

When Vidispine server copies or moves a file between two agent storages, the default is for Vidispine server to read the file from one agent and then write it to the other agent. In the case where the agents actually are able to reach each other, this is obviously quite inefficient, since the data is streamed through Vidispine server.

To let Vidispine server send a transfer job to the agent which hosts the source file, which then sends the file directly to the receiving agent. To enable this you configure both agents with the same value of the agent property `agentGroup`.

The destination URI, where the agent will try to send its file, will as default be the *uri* of the receiving agent, as seen at *GET /vxa/(uuid)*. For example:

```
<VXADocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <uuid>aa4a7ef6-087c-4003-82fb-983c0e91d9c3</uuid>
    <name>Test agent</name>
    <uri>http://localhost:57893/</uri>
    <agentGroup>office-vsa-group</agentGroup>
    ...
</VXADocument>
```

However, in many cases that URI might not be a URI that the first agent can reach, for example if the agent is connected through SSH (then the URI typically is something like: http://localhost:5678/). To overcome this the agents can set the agent property `externalUri` to an URI that the agent can be reached at. This may be used in conjunction with the property: `bindAddressV4` and/or `bindAddressV6`.

### Examples

Two agents are one the same network and connect directly to Vidispine server, we only need to set `agentGroup` in each agents configuration file to the same value:

```
uuid=aa4a7ef6-087c-4003-82fb-983c0e91d9c3
agentGroup=office-vsa-group
...
```

```
uuid=e5db8d36-470c-44fa-8499-967537ddae6a
agentGroup=office-vsa-group
...
```

One agent is connecting to Vidispine server using SSH, we then need to set the `externalUri` property for that agent:

```
uuid=aa4a7ef6-087c-4003-82fb-983c0e91d9c3
agentGroup=office-vsa-group
...
```

```
uuid=e5db8d36-470c-44fa-8499-967537ddae6a
agentGroup=office-vsa-group
externalUri=http://10.0.0.23:8090/
...
```

## 8.5.7 Port forwarding service

New in version 5.1.

It is possible to set up a port forward service, using the already existing connection to Vidispine, for the VSA. This will create a secure channel using remote forwarding. This is done by specifying an ID for the service and the URL and port that this service will try to reach. The agent needs to be configured as such; `port.forward.<id>=<scheme>://<host>:<port>` where <id> needs to be an integer. It is possible for a single VSA to have multiple port forwarding services enabled.

For example:

```
port.forward.1=ldap://someldapserver.com:389
port.forward.2=ldaps://anotherldapserver.com:636
```

after the VSA have connected to Vidispine, the vxa resource will report:

```
GET /vxa HTTP/1.1
```

```
<VXAListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   ..
   <vxa>
     <uuid>e5817fdb-9deb-4f25-a689-72349a78407a</uuid>
     ..
     <forwardService>
       <id>1</id>
       <uri>ldap://examplevshost:40275</uri>
     </forwardService>
     <forwardService>
       <id>2</id>
       <uri>ldaps://examplevshost:43741</uri>
     </forwardService>
   </vxa>
   ..
</VXAListDocument>
```

The example above would be port forwarding for *LDAP authentication*.

New in version 21.3.

For a HTTP connection via VSA, it is recommended to use the VSA as a HTTP proxy instead of forwarding individual ports. For more information about this, see *Proxying HTTP connection via a VSA*.

## 8.6 Vidinet services

Vidinet services can be registered as resources and then be used directly by Vidispine, if supported natively, or from custom JavaScript steps.

### 8.6.1 Adding a service

Add a Vidinet service by creating a new `vidinet` *resource*. The resource document may differ for different services, so please refer to the vidinet dashboard (http://www.vidinet.net/) for more information on how the service should be defined.

```
POST /resource/
Content-Type: application/xml

<?xml version="1.0"?>
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <vidinet>
    <url>http://transcoder.example.com:8888/</url>
    <endpoint>http://transcoder.example.com:8888/</endpoint>
    <type>TRANSCODER</type>
  </vidinet>
</ResourceDocument>
```

Vidispine checks the status of services continuously in the background. As such, if the configuration is correct you will see that the transcoder shows up as `ONLINE` in a few seconds.

```
GET /resource/vidinet/VX-8
```

```
<?xml version="1.0"?>
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-8</id>
  <vidinet>
    <url>http://transcoder.example.com:8888/</url>
    <endpoint>http://transcoder.example.com:8888/</endpoint>
    <type>TRANSCODER</type>
    <state>ONLINE</state>
  </vidinet>
</ResourceDocument>
```

### 8.6.2 Configuring a service

New in version 21.3.

Some VidiNet services require configuration before they can be used. Such as service specific metadata-fields, shape-tags or task definitions. The required configuration can be applied automatically by invoking the configure endpoint of the VidiNet resource. Invoking the endpoint will pull the configuration from VidiNet and apply it. A pre-check can be performed to inspect what will happen before applying the actual configuration:

```
GET /resource/vidinet/VX-1/configuration/pre-check
Content-Type: application/xml
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceConfigurationResultDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <serviceName>MyVidiNetService</serviceName>
  <serviceId>bac7d4ac-69a8-4b52-b421-79f4dfb3177f</serviceId>
  <configurationVersion>1.0.0</configurationVersion>
  <preCheck>true</preCheck>
  <executedSteps>
      <order>1</order>
      <description>Create/Update metadata-field: vidinet_service_metadata_field</
↪description>
      <resource>/metadata-field/vidinet_service_metadata_field</resource>
      <result>The API call will be executed.</result>
      <success>true</success>
  </executedSteps>
</ServiceConfigurationResultDocument>
```

After inspecting the results of the pre-check, the configuration can be applied with:

```
PUT /resource/vidinet/VX-1/configuration
Content-Type: application/xml
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ServiceConfigurationResultDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <serviceName>MyVidiNetService</serviceName>
  <serviceId>bac7d4ac-69a8-4b52-b421-79f4dfb3177f</serviceId>
  <configurationVersion>1.0.0</configurationVersion>
  <preCheck>false</preCheck>
  <executedSteps>
      <order>1</order>
      <description>Create/Update metadata-field: vidinet_service_metadata_field</
↪description>
```

```
      <resource>/metadata-field/vidinet_service_metadata_field</resource>
      <result>The API call was executed successfully.</result>
      <success>true</success>
  </executedSteps>
</ServiceConfigurationResultDocument>
```

### 8.6.3 Import using Vidinet

When importing files the import job uses the Vidispine transcoder to detect the type of media that is being imported, and if requested, to transcode the imported media.

To use Vidinet transcoder service instead of a local transcoder on import, specify the Vidinet `TRANSCODER` resource when initiating the import. For example:

```
POST /import?uri=file:///srv/testdata/sample.mov&tag=__mp4&resourceId=VX-8
```

After the file has been transferred to a storage, the file will be media checked and transcoded using Vidinet. Once a media check or transcode has been requested from Vidinet, the state of the job will change to `VIDINET_JOB`, and the job will no longer occupy a job slot, until Vidinet has completed and the job will resume.

Make sure that files are imported to or exist on a storage that is compatible with the Vidinet service in question. This is typically either an S3 bucket or an Azure blob storage, see the Vidinet service documentation for more detail.

### 8.6.4 Transcoding using Vidinet

If a Vidinet `TRANSCODER` resource is specified when initiating an item transcode, then the transcode will be performed using that Vidinet service. For example:

```
POST /item/VX-74/transcode?tag=__mp4&resourceId=VX-8
```

The transcode job will execute as normal, but the transcode will be handed off to Vidinet instead of being sent to a local transcoder. Once this happens the state of the job will change to `VIDINET_JOB`, and no longer occupy a job slot, until Vidinet has completed the transcode.

#### Cost estimation

To retrieve the estimated cost of performing the above transcode using Vidinet, the *cost API* can be used. Simply prefix the path with `cost/` and execute the request:

```
POST /cost/item/VX-74/transcode?tag=__mp4&resourceId=VX-8
```

```
<CostEstimateDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>dGVzdA==</id>
  <url>http://localhost:8080/API/cost/estimate/dGVzdA==</url>
</CostEstimateDocument>
```

The estimate may not be immediately available, in which case the estimate will be shown as pending.

```
GET http://localhost:8080/API/cost/estimate/dGVzdA==
```

```
<CostEstimateDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>dGVzdA==</id>
  <url>http://localhost:8080/API/cost/estimate/dGVzdA==</url>
  <state>PENDING</state>
  <service>
    <resource>VX-8</resource>
```

```
    <type>TRANSCODER</type>
    <state>ONLINE</state>
  </service>
</CostEstimateDocument>
```

Once the cost has been estimated:

```
GET http://localhost:8080/API/cost/estimate/dGVzdA==
```

```
<CostEstimateDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>dGVzdA==</id>
  <url>http://localhost:8080/API/cost/estimate/dGVzdA==</url>
  <state>FINISHED</state>
  <service>
    <resource>VX-8</resource>
    <type>TRANSCODER</type>
    <state>ONLINE</state>
    <cost unit="USD">1.2</cost>
  </service>
</CostEstimateDocument>
```

### 8.6.5 Quality control using Vidinet

Quality control using Vidinet services can be performed by specifying a Vidinet QC resource when starting a *shape analysis job*. For more information on how to analyze using a Vidinet service, please refer to the Vidinet service documentation.

```
POST /item/VX-74/shape/VX-79/analyze?resourceId=VX-3&jobmetadata=template%3DQuality
 ↪%20Test
Content-Type: application/xml

<AnalyzeJobDocument xmlns="http://xml.vidispine.com/schema/vidispine"/>
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-345</jobId>
  <user>admin</user>
  <started>2017-09-08T14:59:49.131Z</started>
  <status>READY</status>
  <type>ANALYZE</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

Once the job has finished, the result of the analysis can be found in the *bulky metadata* of the shape.

A cost estimate can be retrieved, just like for transcodes, using the *cost API*.

### 8.6.6 Using Vidinet services from JavaScript

Vidinet services that are not natively supported can be used from JavaScript, for example by creating a *custom job* with one or more steps that interact with the Vidinet service using the *Vidinet JavaScript functions*.

For example:

```
POST /task-definition/jobtype/MYCOMPANY_CUSTOM_VIDINET_JOB?id=26000
```

```
PUT  /task-definition/jobtype/MYCOMPANY_CUSTOM_VIDINET_JOB/step/100
Content-Type: application/xml

<TaskDefinitionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <description>A custom JavaScript step</description>
  <script><![CDATA[
var item = ...;
var instruction = "...";

job.vidinetJob("TEST", instruction, {
    item: itemId
});
]]></script>
  <step>100</step>
  <dependency>
    <previous>false</previous>
    <allPrevious>true</allPrevious>
  </dependency>
  <jobType>MYCOMPANY_CUSTOM_VIDINET_JOB</jobType>
  <critical>false</critical>
</TaskDefinitionDocument>
```

For more information on how to execute jobs for a service in Vidinet, please refer to the Vidinet service documentation.

### 8.6.7 Transcoding using AWS Elemental MediaConvert

*The AWS Elemental MediaConvert integration is currently in developer preview. This means that syntax may change somewhat for the final implementation.*

New in version 4.15.

You can use Elemental MediaConvert to transcode your files using Vidinet. To start with you need to buy the service in Vidinet and add it to your Vidispine server instance. please refer to the Vidinet service documentation on how to do that.

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-1</id>
  <vidinet>
    <url>vidinet://e1423727-...:...@42a73b4c-8974-4402-a237-17b80bd11350</url>
    <name>My AWS MediaConvert</name>
    <endpoint>https://services.vidinet.net</endpoint>
    <type>ELEMENTAL_MEDIACONVERT</type>
    <state>ONLINE</state>
    <scheme>s3</scheme>
  </vidinet>
</ResourceDocument>
```

You then need a new shape-tag with the new `mediaconvert` element. To install the system default shape-tags that use Elastic MediaConvert you call:

```
PUT /APIinit/preset-mediaconvert-templates
```

Once the `vidinet` resource is in place and a shape-tag contains the `mediaconvert` element you can use it as any other shape-tag for transcoding.

```
 <TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <description>
     BROADCAST, XDCAM, MXF, MPEG2 HD422, WAV, 16x9 DAR, 1920x1080p, 23.98 Hz, 50 Mbps␣
↪CBR
```

```
  </description>
  <name>
    __mediaconvert_Broadcast_Xdcam_Mxf_Mpeg2_Wav_16x9_1920x1080p_24Hz_50Mbps
  </name>
  <audio/>
  <video/>
  <mediaconvert>
    <outputSetting>
        { "Type": "SYSTEM", "Category": "BROADCAST-XDCAM", ... }
    </outputSetting>
  </mediaconvert>
</TranscodePresetDocument>
```

Transcoding using this preset would then cause the transcode to be executed using the Vidinet Elemental MediaConvert service:

```
POST /item/VX-46/transcode?tag=__mediaconvert_Broadcast_Xdcam_Mxf_Mpeg2_Wav_16x9_
↪1920x1080p_24Hz_50Mbps
```

The following requirements apply when using MediaConvert:

- The input and output storages needs to be S3 buckets.
- The buckets must be accessible to the AWS Elemental MediaConvert service as detailed in the Vidinet service documentation.

### 8.6.8 Transcoding using Bitmovin

New in version 5.0.

You can use the Bitmovin service in Vidinet to transcode your files. To start with you need to buy the service in Vidinet and add it to your Vidispine server instance. Please refer to the Vidinet service documentation on how to do that.

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-1</id>
  <vidinet>
    <url>vidinet://g6d23345-...:...@4b916a24-393d-4ff6-85ac-76e3fb082dd9</url>
    <name>My Bitmovin transcoder</name>
    <endpoint>https://services.vidinet.net</endpoint>
    <type>BITMOVIN</type>
    <state>ONLINE</state>
    <scheme>s3</scheme>
  </vidinet>
</ResourceDocument>
```

Once the `vidinet` resource is in place you can use any normal shape-tag for transcoding. Please see the Vidinet service documentation for current restrictions, as they are subject to change with upcoming updates to the Vidinet system.

The following requirements apply when using Bitmovin:

- The input and output storages needs to be S3 buckets.
- The buckets must be accessible to the Bitmovin service as detailed in the Vidinet service documentation.

### 8.6.9 Analyzing using Vidinet Cognitive Services

New in version 5.0.

You can use Vidinet Cognitive Services to analyze your files and populate them with cognitive metadata. To start you need to buy a cognitive service in the Vidinet store and add it to your Vidispine server instance. Please refer to the Vidinet service documentation on how to do that.

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-1</id>
  <vidinet>
    <url>vidinet://e1423727-...:...@42a73b4c-8974-4402-a237-17b80bd11350</url>
    <name>My AWS Rekognition service</name>
    <endpoint>https://services.vidinet.net</endpoint>
    <type>COGNITIVE_SERVICES</type>
    <state>ONLINE</state>
    <scheme>s3</scheme>
  </vidinet>
</ResourceDocument>
```

Once the Vidinet resource is in place you can trigger an analysis on an item:

```
POST /item/VX-46/analyze?resourceId=VX-1
```

```
<AnalyzeJobDocument xmlns="http://xml.vidispine.com/schema/vidispine"/>
```

Or, on a specific shape:

```
POST /item/VX-46/shape/VX-47/analyze?resourceId=VX-1
```

```
<AnalyzeJobDocument xmlns="http://xml.vidispine.com/schema/vidispine"/>
```

The following requirements apply when using AWS based Cognitive Services:

- The media storage needs to be an S3 bucket.
- The buckets must be accessible to the AWS Cognitive service as detailed in the Vidinet service documentation.

### 8.6.10 Training custom models using VidiNet Cognitive Services

New in version 21.3.

You can use VidiNet Cognitive Services to train your own models which can be used for detection in an analyze job. Please refer to the knowledge base (http://www.vidispine.com/partner/knowledge-forum-support) for detailed instructions on how to get started with custom training.

### 8.6.11 Creating a highlight reel using Nablet Shrynk

New in version 5.4.

Nablet Shrynk is an AI technology that analyses the content of a video file and assigns an interest factor for each frame in the video. Once the analysis has been completed a highlight reel can be rendered for any desired output length. In order to perform the analysis, you need a GPU enabled transcoding service in Vidinet.

Analyze a shape with default parameters:

```
POST /item/VX-123/shape/VX-456/analyze
Content-Type: application/xml

<AnalyzeJobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <highlighter>
    <model>football</model>
```

```
  </highlighter>
</AnalyzeJobDocument>
```

When the analysis has finished, start a render job:

```
POST /item/VX-123/shape/VX-456/highlight-render?tag=__mp4&duration=300
```

### 8.6.12 Using Nablet Heightscreen to crop a video into portrait mode

New in version 5.4.

Oftentimes content is filmed in landscape mode, which is not suitable for every device. Nablet Heighscreen uses AI to determine the areas of highest interest in the video. Once an analysis has been completed for a specific aspect ratio, a job can be started to render the asset in portrait mode. A GPU enabled transcoder service is needed in order to perform the analysis.

```
POST /item/VX-123/shape/VX-456/analyze
Content-Type: application/xml

<AnalyzeJobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <smartcrop>
    <aspect>9:16</aspect>
  </smartcrop>
</AnalyzeJobDocument>
```

When the analysis has finished, start a render job:

```
POST /item/VX-123/shape/VX-456/smartcrop-render?tag=__mp4&aspect=9_16
```

### 8.6.13 Using Interra Baton to perform quality control on your material

New in version 5.6.

Baton QC is an industry standard for performing quality control on media content. It supports the basic checks for black frames, freezes all the way up to advanced features such as PSE checks. We have built an integration available in VidiNet, which means you only have to add the service in your VidiNet dashboard. The service is automatically attached to your VidiCore instance (provided it is running in VidiNet), and we also attach some select Baton test plans to get you started. Each test plan is added to VidiCore as an analyze preset. The following test plans are provided out of the box:

- Generic - Has all the basic checks enabled, such as black/freeze frames, blockiness, RGB color gamut and signal level checks.
- GenericHDR - Same as the Generic plan, but made for HDR content.
- GenericPSE - Same as the Generic plan, but with PSE checks enabled.
- XDCAM_HD_422_MXF_1080i50 - Has specific checks for validating XDCAM HD content.
- AS-11_UK_DPP_HD - Has all the checks for validating AS-11 UK DPP files.

To start a validation, we use a standard ANALYZE job, and provide a Baton test plan as the preset:

```
POST /item/VX-123/shape/VX-456/analyze?preset=Generic
```

VidiCore will see that the preset is a Baton test plan and delegate the job to Baton instance running in the cloud. Once the job has finished, the Baton reports are stored in the bulky metadata of the shape. We can then extract the files by asking for the *bulky metadata as a file*. Looking at the resulting bulky metadata, we get the following result:

```
GET /item/VX-123/shape/VX-456/metadata/bulky
```

```
<?xml version="1.0"?>
<URIListDocument>
    <uri>baton_error_Generic</uri>
    <uri>baton_report_files</uri>
    <uri>baton_summary_Generic</uri>
</URIListDocument>
```

The *baton_error_Generic* contains a parsed version of the Baton XML report, which can be used to display the errors in a player for example. The *baton_summary_Generic* key contains a summary of the number of errors, warnings and informational messages from the analysis. Finally, the *baton_report_files* contain the binary data from the PDF and XML reports. If we look at what this key contains, we can see the following:

```
GET /item/VX-123/shape/VX-456/metadata/bulky/baton_report_files
```

```
<?xml version="1.0"?>
<BulkyMetadataDocument id="VX-456">
    <field start="-INF" end="+INF" itemTrack="">
        <key>baton_report_files</key>
        <maps>
            <map>
                <entry key="filename">baton_VX-123_VX-456_Generic.xml</entry>
                <entry key="content">...</entry>
                <entry key="type">xml</entry>
                <entry key="test-plan">Generic</entry>
                <entry key="created">2021-04-09T13:19:48.164Z</entry>
            </map>
            <map>
                <entry key="filename">baton_VX-123_VX-456_Generic.pdf</entry>
                <entry key="content">...</entry>
                <entry key="type">pdf</entry>
                <entry key="test-plan">Generic</entry>
                <entry key="created">2021-04-09T13:19:48.405Z</entry>
            </map>
        </maps>
    </field>
</BulkyMetadataDocumen
```

The actual file data has been removed from the above snippet to save space. Using the bulky-data-as-file endpoint we can now get the actual fle content using the following call:

```
GET /item/VX-123/shape/VX-456/metadata/bulky/baton_report_files/as-file?
↪extraMapValues=type=pdf
```

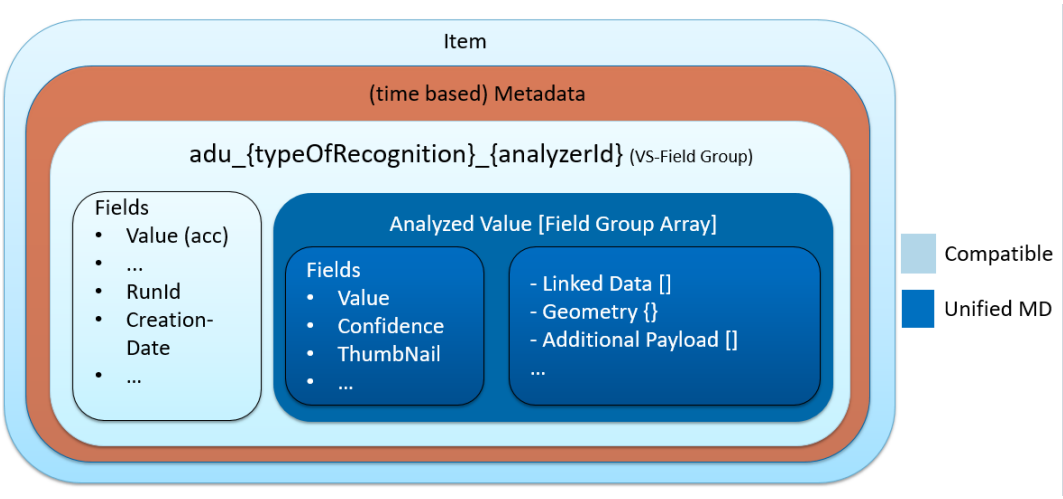
The *extraMapValues* is used to filter which *<map>* to extract the file from. In this case we use the *type* entry and specify that we want the map which has a "type" entry with a value of "pdf".

## 8.7 Analyzed Data Unit (ADU)

New in version 5.0.

The analyzed data unit (ADU) is a type of metadata field group containing cognitive analysis metadata, for example transcript data, celebrity detection, or content moderation. This metadata is gathered from one or more cognitive service providers and transformed into a standardized format when using the *Vidinet Cognitive Services*. This standardized format conforms to the *item metadata model*.

Every ADU item metadata field group and fields start with the prefix `adu_`. As this prefix is added by Vidinet Cognitive Services, it should not be used for other metadata purposes. These structures must not be modified.




At the top level, the field group is named as `adu_<typeOfRecognition>_<analyzerId>`. In addition to the standard metadata fields, it contains metadata fields about the cognitive service provider being used (here referred to as `analyzerId`) and what type of cognitive analysis was performed (here referred to as `typeOfRecognition`).

Furthermore, the cognitive analysis metadata itself is stored as field groups inside the top level field group, with the name of `adu_av_analyzedValue`. These field groups can contain these fields:

| Field Name | Value |
|---|---|
| `adu_av_value` | The metadata that has been identified |
| `adu_av_confidence` | (Optional) A confidence value between 0 and 1 |
| `adu_av_description` | (Optional) Description of metadata |
| `adu_av_thumbnailUrl` | A thumbnail representing the metadata, if available from Vidinet Cognitive Services |

### 8.7.1 Example ADU

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="184@PAL" end="1397@PAL">
    <group>
      <name>adu_label_AWSVideoRekognitionAnalyzer</name>
      <field>
        <name>adu_analyzerId</name>
        <value>AWSVideoRekognitionAnalyzer</value>
      </field>
      <field>
        <name>adu_analysisType</name>
        <value>label</value>
      </field>
      <field>
        <name>adu_creationDate</name>
        <value>2019-07-10T11:08:34</value>
      </field>
      <field>
        <name>adu_analysisMonitorId</name>
        <value>9fafb662-7399-4af5-b50e-22530b4e6948</value>
      </field>
      <field>
        <name>adu_value</name>
```

```
      <value>Apparel, Clothing, Coat, Overcoat, Suit</value>
    </field>
    <group>
      <name>adu_av_analyzedValue</name>
      <field>
        <name>adu_av_value</name>
        <value>Apparel</value>
      </field>
      <field>
        <name>adu_av_confidence</name>
        <value>0.92</value>
      </field>
      <field>
        <name>adu_av_description</name>
      </field>
      <field>
        <name>adu_av_thumbnailUrl</name>
        <value>https://via.placeholder.com/160x90.jpg?text=placeholder</value>
      </field>
    </group>
    <group>
      <name>adu_av_analyzedValue</name>
      <field>
        <name>adu_av_value</name>
        <value>Clothing</value>
      </field>
      <field>
        <name>adu_av_confidence</name>
        <value>0.92</value>
      </field>
      <field>
        <name>adu_av_description</name>
      </field>
      <field>
        <name>adu_av_thumbnailUrl</name>
        <value>https://via.placeholder.com/160x90.jpg?text=placeholder</value>
      </field>
    </group>
    <group>
      <name>adu_av_analyzedValue</name>
      <field>
        <name>adu_av_value</name>
        <value>Coat</value>
      </field>
      <field>
        <name>adu_av_confidence</name>
        <value>0.92</value>
      </field>
      <field>
        <name>adu_av_description</name>
      </field>
      <field>
        <name>adu_av_thumbnailUrl</name>
        <value>https://via.placeholder.com/160x90.jpg?text=placeholder</value>
      </field>
    </group>
    <group>
      <name>adu_av_analyzedValue</name>
```

```
        <field>
          <name>adu_av_value</name>
          <value>Overcoat</value>
        </field>
        <field>
          <name>adu_av_confidence</name>
          <value>0.92</value>
        </field>
        <field>
          <name>adu_av_description</name>
        </field>
        <field>
          <name>adu_av_thumbnailUrl</name>
          <value>https://via.placeholder.com/160x90.jpg?text=placeholder</value>
        </field>
      </group>
      <group>
        <name>adu_av_analyzedValue</name>
        <field>
          <name>adu_av_value</name>
          <value>Suit</value>
        </field>
        <field>
          <name>adu_av_confidence</name>
          <value>0.92</value>
        </field>
        <field>
          <name>adu_av_description</name>
        </field>
        <field>
          <name>adu_av_thumbnailUrl</name>
          <value>https://via.placeholder.com/160x90.jpg?text=placeholder</value>
        </field>
      </group>
    </group>
  </timespan>
  <timespan>...</timespan>
</MetadataDocument>
```

## 8.8 Callback location resources

**Note:** Currently, this feature only works together with ANALYZE jobs

A callback location resource points to a location where VidiCore can expect to find callback documents which contain scripts that should be executed as part of a job.

Scripts are JavaScripts and are executed by the built in JavaScript engine. See *JavaScript*.

Currently only S3-buckets are supported as callback locations.

### 8.8.1 Adding a callback location resource

Add a callback location resource using *POST /resource*.

```
POST /resource
Content-Type: application/xml

<?xml version="1.0"?>
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <callback>
    <uri>s3://name:pass@example-bucket/folder1</uri>
  </callback>
</ResourceDocument>
```

### 8.8.2 Callback Document format

Callback documents should be placed in the location indicated by the callback location resource you wish to use. They are formatted as such:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CallbackDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <name>Name of callback script</name>
  <description>Description of callback script.</description>
  <script>
    JavaScript here....
  </script>
</CallbackDocument>
```

Note that they must be saved as .xml-files for the callback executor to recognize them.

CHAPTER

NINE

# TIMELINES AND SEQUENCES

## 9.1 Projects and sequences

### 9.1.1 Item sequences

An item can hold a number of sequences, and is then called a sequence item. All sequences will be considered equivalent by Vidispine, that is, that they represent the same logical sequence.

Sequences can also be *imported and exported* to and from common NLE formats.

The non-timed metadata of a sequence item will contain the following fields:

| Field Name | Value |
|---|---|
| `__sequence_size` | The number of sequences that exist for an item. |
| `__sequence` | The format of a sequence that exist for an item. |

### 9.1.2 Projects and project versions

A project is a special type of *collection* that contains a number of project versions. A project version is a collection that contains the *items* and *sequences* that together represent a specific version. As both project and project versions are ordinary collections it means that all existing collection operations can be used, for example editing project *metadata*.

Projects can also be *imported and exported* to and from common NLE formats.

---

**Note:** Projects and project versions are read-only and cannot be altered by manually adding or removing child items or collections.

---

For a project version it is possible to store the original document representing the project, the Final Cut Pro XML for example, as well as any additional representations, here called Project Version Definitions. Each representation is stored as binary data, and is identified by a format identifier (e.g. `finalcut`.)

Any string can be used as the format identifier, except the following which are reserved by Vidispine, but may be used as long as the content matches.

| Identifier | Content | Description |
|---|---|---|
| `finalcut` | `application/final-cut-pro` | Final Cut Pro 7 XML |
| `finalcut-x` | `application/final-cut-pro-x` | Final Cut Pro X XML |
| `aaf` | `application/aaf` | AAF |
| `fabric` | `application/fabric` | Fabric CEMS |
| `vidispine` | `application/x-vidispine` | *SequenceType* |

#### Example

For a project named “Unnamed project”:

```
<timespan start="-INF" end="+INF">
  <field>
    <name>__type</name>
    <value>project</value>
  </field>
  <field>
    <name>__project_name</name>
    <value>Unnamed Projekt</value>
  </field>
  ...
</timespan>
```

For a project version with a single Final Cut Pro representation:

```
<timespan start="-INF" end="+INF">
  <field>
    <name>__type</name>
    <value>projectVersion</value>
  </field>
  <field>
    <name>__project_version</name>
    <value>finalcut</value>
  </field>
  ...
</timespan>
```

#### Metadata

Projects and project version collections contains additional (non-timed) metadata that may be useful when searching for collection.

| Field Name | Value |
|---|---|
| `__type` | `project` for project collections.<br>`projectVersion` for project version collections. |
| `__project_name` | The name of the project. |
| `__project_version` | The format of the definitions that have been stored for a project version. |

### 9.1.3 Project and sequence import and export

This page describes how to import and export *projects* and *sequences* from NLEs such as Final Cut Pro and Avid Media Composer.

#### Inspecting a project file

Before a project or sequence can be imported, the project file has to be inspected in order to find out which clips already exist in Vidispine as items, and which must first be imported.

The input should be an essence mappings document, which is also used for project and sequence import. It is required so that Vidispine can identify the items and files referenced by the input project file. The document can specify:

- The SHA-1 hash of a file. The response will then contain all items and shapes that reference that specific file.
- The item corresponding to a specific asset. Can be used after a previously unknown asset has been imported and the correct item is known. If the item has multiple shapes then the shape id must be specified as well.
- If a storage has been locally mounted on the client, then a storage mapping containing the id of the storage and the local path can be given. This will only be used if the input file references files by path.

### Example

```
POST /collection/project/inspect?uri=file:///home/maria/sequence.xml&amp;type=finalcut
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EssenceMappingDocument xmlns="http://xml.vidispine.com/schema/vidispine">
</EssenceMappingDocument>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProjectFileDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <location>file:///home/maria/sequence.xml</location>
  <asset>
    <id>urn:uuid:8CED8AFE-1A67-4632-AB57-D5F5B1E0BC49</id>
    <name>Sequence 1</name>
    <type>sequence</type>
    <status>unknown</status>
  </asset>
  <asset>
    <id>urn:uuid:FCAD0878-7129-43DA-A8A0-696590EFE4DA</id>
    <name>Sample Clip B</name>
    <type>clip</type>
    <status>unknown</status>
    <file>
      <path>file://localhost/Users/maria/Sample%20Clip%20B.mov</path>
    </file>
  </asset>
  <asset>
    <id>urn:uuid:76BE320F-48E0-47A5-A076-227158C50024</id>
    <name>Clip A</name>
    <type>clip</type>
    <status>unknown</status>
    <file>
      <path>file://localhost/Users/maria/Movies/Vidispine/VX-1.mov</path>
    </file>
  </asset>
</ProjectFileDocument>
```

With the SHA-1 hash provided for all of the files:

```
POST /collection/project/inspect?uri=file:///home/maria/sequence.xml&amp;type=finalcut
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EssenceMappingDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <file path="file://localhost/Users/maria/Movies/Vidispine/VX-1.mov" hash=
↪"7b8d6ffe1ea468800578d6b7d4a09b012c461569"/>
  <file path="file://localhost/Users/maria/Sample%20Clip%20B.mov" hash=
↪"c7cfc97a9cf6634ad94766c0c4b0789cd86bcc33"/>
</EssenceMappingDocument>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProjectFileDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <location>file:///home/maria/sequence.xml</location>
  <asset>
    <id>urn:uuid:8CED8AFE-1A67-4632-AB57-D5F5B1E0BC49</id>
    <name>Sequence 1</name>
    <type>sequence</type>
```

```
    <status>unknown</status>
  </asset>
  <asset>
    <id>urn:uuid:FCAD0878-7129-43DA-A8A0-696590EFE4DA</id>
    <name>Sample Clip B</name>
    <type>clip</type>
    <status>unknown</status>
    <file>
      <path>file://localhost/Users/maria/Sample%20Clip%20B.mov</path>
    </file>
  </asset>
  <asset>
    <id>urn:uuid:76BE320F-48E0-47A5-A076-227158C50024</id>
    <name>Clip A</name>
    <type>clip</type>
    <item id="VX-1" match="file" permission="OWNER"/>
    <file>
      <path>file://localhost/Users/maria/Movies/Vidispine/VX-1.mov</path>
      <hash>7b8d6ffe1ea468800578d6b7d4a09b012c461569</hash>
      <file>
        <id>VX-1</id>
        <path>VX-1.mov</path>
        <uri>file:///mnt/storage/Vidispine/VX-1.mov</uri>
        <state>CLOSED</state>
        <size>30346173</size>
        <timestamp>2011-10-13T07:41:48.053+02:00</timestamp>
        <refreshFlag>727</refreshFlag>
        <storage>VX-1</storage>
        <item>
          <id>VX-1</id>
          <shape>
            <id>VX-1</id>
            <component>
              <id>VX-1</id>
            </component>
            <component>
              <id>VX-1</id>
            </component>
            <component>
              <id>VX-1</id>
            </component>
            <component>
              <id>VX-1</id>
            </component>
            <component>
              <id>VX-1</id>
            </component>
          </shape>
        </item>
      </file>
    </file>
  </asset>
</ProjectFileDocument>
```

After the new asset has been imported into Vidispine:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProjectFileDocument xmlns="http://xml.vidispine.com/schema/vidispine">
```

```
<location>file:///home/maria/sequence.xml</location>
<asset>
  <id>urn:uuid:8CED8AFE-1A67-4632-AB57-D5F5B1E0BC49</id>
  <name>Sequence 1</name>
  <type>sequence</type>
  <status>unknown</status>
</asset>
<asset>
  <id>urn:uuid:FCAD0878-7129-43DA-A8A0-696590EFE4DA</id>
  <name>Sample Clip B</name>
  <type>clip</type>
  <item id="VX-2" match="file" permission="OWNER"/>
  <file>
    <path>file://localhost/Users/maria/Sample%20Clip%20B.mov</path>
    <hash>c7cfc97a9cf6634ad94766c0c4b0789cd86bcc33</hash>
    <file>
      <id>VX-2</id>
      <path>VX-2.mov</path>
      <uri>file:///mnt/storage/Vidispine/VX-2.mov</uri>
      <state>CLOSED</state>
      <size>30346173</size>
      <timestamp>2011-10-13T07:42:48.178+02:00</timestamp>
      <refreshFlag>727</refreshFlag>
      <storage>VX-1</storage>
      <item>
        <id>VX-2</id>
        <shape>
          <id>VX-2</id>
          <component>
            <id>VX-2</id>
          </component>
          <component>
            <id>VX-2</id>
          </component>
          <component>
            <id>VX-2</id>
          </component>
        </shape>
      </item>
    </file>
  </file>
</asset>
<asset>
  <id>urn:uuid:76BE320F-48E0-47A5-A076-227158C50024</id>
  <name>Clip A</name>
  <type>clip</type>
  <item id="VX-1" match="file" permission="OWNER"/>
  <file>
    <path>file://localhost/Users/maria/Movies/Vidispine/VX-1.mov</path>
    <hash>7b8d6ffe1ea468800578d6b7d4a09b012c461569</hash>
    <file>
      <id>VX-1</id>
      <path>VX-1.mov</path>
      <uri>file:///mnt/storage/Vidispine/VX-1.mov</uri>
      <state>CLOSED</state>
      <size>30346173</size>
      <timestamp>2011-10-13T07:41:48.053+02:00</timestamp>
      <refreshFlag>727</refreshFlag>
```

```
        <storage>VX-1</storage>
        <item>
          <id>VX-1</id>
          <shape>
            <id>VX-1</id>
            <component>
              <id>VX-1</id>
            </component>
            <component>
              <id>VX-1</id>
            </component>
            <component>
              <id>VX-1</id>
            </component>
            <component>
              <id>VX-1</id>
            </component>
            <component>
              <id>VX-1</id>
            </component>
            <component>
              <id>VX-1</id>
            </component>
          </shape>
        </item>
      </file>
    </file>
  </asset>
</ProjectFileDocument>
```

## 9.2 Sequences definitions

### 9.2.1 SequenceDocument

SequenceDocument (XML complex type SequenceType) is a simple format for describing a sequence, with a model similar to sequences in the Final Cut Pro XML interchange format.

#### Structure

A sequence consists of a number of audio and/or video tracks where each track may consist of one of more segments (to make clips appear edge-to-edge in the generated timeline). Each segment has a position in the timeline (`in` and `out`) and references a specific interval and track of an item (`item`, `sourceTrack` (1-based), `sourceIn` and `sourceOut`.)

For video the `sourceTrack` element specifies the **n** th video track that should be included. For audio it specifies a specific channel in an audio track. For example, for media with two audio streams each with two audio channels, `sourceTrack=3` would specific the first channel in the second audio stream.

The elements `in` / `out` and `sourceIn` / `sourceOut` corresponds to the Final Cut Pro XML elements `in` / `out` and `start` / `end` respectively. See Timing Values (http://developer.apple.com/library/mac/#documentation/AppleApplications/Reference/FinalCutPro_XML/Topics/Topics.html#//apple_ref/doc/uid/TP30001154-CH294-SW12) for more information.

#### Example

A sequence with a single video track with one second of video from the item with id `VX-1`.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SequenceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
```

```
  <track>
    <audio>false</audio>
    <segment>
      <item>VX-1</item>
      <sourceTrack>1</sourceTrack>
      <in>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </in>
      <out>
        <samples>25</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </out>
      <sourceIn>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceIn>
      <sourceOut>
        <samples>25</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceOut>
    </segment>
  </track>
</SequenceDocument>
```

If the item has 10 minutes of video and stereo audio, it could be included in a sequence like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SequenceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <track>
    <audio>false</audio>
    <segment>
      <item>VX-1</item>
      <sourceTrack>1</sourceTrack>
      <in>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </in>
      <out>
        <samples>15000</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
```

```
        </timeBase>
      </out>
      <sourceIn>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceIn>
      <sourceOut>
        <samples>15000</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceOut>
    </segment>
  </track>
  <track>
    <audio>true</audio>
    <segment>
      <item>VX-1</item>
      <sourceTrack>1</sourceTrack>
      <in>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </in>
      <out>
        <samples>15000</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </out>
      <sourceIn>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceIn>
      <sourceOut>
        <samples>15000</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceOut>
    </segment>
  </track>
  <track>
    <audio>true</audio>
    <segment>
      <item>VX-1</item>
      <sourceTrack>2</sourceTrack>
```

```
      <in>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </in>
      <out>
        <samples>15000</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </out>
      <sourceIn>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceIn>
      <sourceOut>
        <samples>15000</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceOut>
    </segment>
  </track>
</SequenceDocument>
```

A two seconds long video track sequence made from the first second of video from items VX-1 and VX-2 using multiple segments on one track.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SequenceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <track>
    <audio>false</audio>
    <segment>
      <item>VX-1</item>
      <sourceTrack>1</sourceTrack>
      <in>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </in>
      <out>
        <samples>25</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </out>
      <sourceIn>
        <samples>0</samples>
```

```
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceIn>
      <sourceOut>
        <samples>25</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceOut>
    </segment>
    <segment>
      <item>VX-2</item>
      <sourceTrack>1</sourceTrack>
      <in>
        <samples>25</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </in>
      <out>
        <samples>50</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </out>
      <sourceIn>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceIn>
      <sourceOut>
        <samples>25</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceOut>
    </segment>
  </track>
</SequenceDocument>
```

### Effects

The table below describes the effects that can be added to segments in a sequence.

| Effect | Parameter | Range | Description |
|---|---|---|---|
| crop | left | 0.0–1.0 | Percentage to crop from left side of the picture |
| | right | 0.0–1.0 | Percentage to crop from right side of the picture |
| | top | 0.0–1.0 | Percentage to crop from the top |
| | bottom | 0.0–1.0 | Percentage to crop from the bottom |
| position | vert | -1.0–1.0 | Vertical offset in output in percentage. |
| | horiz | -1.0–1.0 | Horizontal offset in output in percentage. |
| scale | scale | 0.0–Inf | Horizontal and vertical scale. |
| rotation | rotation | Inf–Inf | Number of degrees to rotate picture, clockwise, around center. |
| opacity | opacity | 0.0–100.0 | The opacity, from fully transparent (0.0) to fully opaque (100.0). |

Effects are added in the follow way:

```
<segment>
  ...
  <effect name="scale">
    <parameter name="scale" value="50"/>
  </effect>
</segment>
```

Effects can also be applied at specific key frames.

```
<segment>
  ...
  <effect name="scale">
    <parameter name="scale">
      <point position="0" value="0"/>
      <point position="125" value="100"/>
    </parameter>
  </effect>
</segment>
```

Rendering two different videos to one view by scaling and positioning them.

```
<track>
    <audio>false</audio>
    <segment>
        <item>VX-1</item>
        <sourceTrack>1</sourceTrack>
        <in>
            <samples>0</samples>
            <timeBase>
                <numerator>1</numerator>
                <denominator>25</denominator>
            </timeBase>
        </in>
        <out>
            <samples>250</samples>
            <timeBase>
                <numerator>1</numerator>
                <denominator>25</denominator>
            </timeBase>
        </out>
        ...
        <effect name="position">
            <parameter name="horiz" value="-0.25"/>
            <parameter name="vert" value="0.25"/>
        </effect>
```

```
        <effect name="scale">
            <parameter name="scale" value="50"/>
        </effect>
    </segment>
</track>
<track>
    <audio>false</audio>
    <segment>
        <item>VX-2</item>
        <sourceTrack>1</sourceTrack>
        <in>
            <samples>0</samples>
            <timeBase>
                <numerator>1</numerator>
                <denominator>25</denominator>
            </timeBase>
        </in>
        <out>
            <samples>250</samples>
            <timeBase>
                <numerator>1</numerator>
                <denominator>25</denominator>
            </timeBase>
        </out>
        ...
        <effect name="position">
            <parameter name="horiz" value="0.25"/>
            <parameter name="vert" value="0.25"/>
        </effect>
        <effect name="scale">
            <parameter name="scale" value="50"/>
        </effect>
    </segment>
</track>
```

### Transitions

The table below describes the transitions that can be added between segments in video tracks in a sequence. If a transition has a corresponding SMPTE wipe code (http://www.w3.org/TR/2005/REC-SMIL2-20050107/smil-transitions.html#TransitionEffects-Appendix) , then either the transition name or wipe code can be used to select that transition.

| Transition | SMPTE Wipe Code |
|---|---|
| Dissolves | |
| CrossDissolve | - |
| DitherDissolve | - |
| FadeInOutDissolve | - |
| Wipes | |
| BandWipe | - |
| CentreWipe | 21 or 22 |
| CheckerWipe | - |
| InsetWipe | 3, 4, 5 or 6 |
| Iris Wipes | |
| CrossIris | 7 |
| DiamondIris | 102 |
| OvalIris | 119 |
| RectangleIris | 101 |
| StarIris | 128 |

### Example

A sequence with two clips that are transitioned using a star wipe:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SequenceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <track>
    <audio>false</audio>
    <segment>
      <item>VX-1</item>
      <sourceTrack>1</sourceTrack>
      <in>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </in>
      <out>
        <samples>15000</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </out>
      <sourceIn>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceIn>
      <sourceOut>
        <samples>15000</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceOut>
```

```
    </segment>
    <segment>
      <item>VX-1</item>
      <sourceTrack>1</sourceTrack>
      <in>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </in>
      <out>
        <samples>15000</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </out>
      <sourceIn>
        <samples>15000</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceIn>
      <sourceOut>
        <samples>30000</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceOut>
    </segment>
    <transition>
      <in>
        <samples>14975</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </in>
      <out>
        <samples>15025</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </out>
      <transition>StarIris</transition>
    </transition>
  </track>
</SequenceDocument>
```

### Overriding shape tag transcode preset

New in version 5.6.

The SequenceDocument used for rendering a sequence can also override the output settings of the shape tag specified

for the rendition, see the `override` element in SequenceType in *XML Schema* for details.

### Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SequenceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <track>
  ...
  </track>
  <override>
    <audio>
      <mix>
         <input channel="0" gain="1.0"/>
         <input channel="2" gain="1.0"/>
      </mix>
      <mix>
         <input channel="1" gain="1.0"/>
         <input channel="3" gain="1.0"/>
      </mix>
    </audio>
  </override>
</SequenceDocument>
```

### Reference external media on the timeline

New in version 21.3.

The SequenceDocument used for rendering a sequence normally references items that are known to VidiCore, but the user can also reference external video media by using an URI. See the `externalVideoMedia` in SequenceMediaType in *XML Schema* for details. The `externalVideoMedia` element has a couple of mandatory elements and some optional.

| Element | Explanation |
|---|---|
| uri | URI where VidiCore can access the media. |
| format | File format/extension of the media. For example mov or mxf. |
| essenceStreamId | Zero based stream/track id of the stream/track to use in the media. |
| timeBase | Time base of the media, used with samples to calculate the media duration. |
| samples | Number of samples, used with timeBase to calculate the media duration. |
| width | Width of the media. |
| height | Height of the media. |
| pixelAspectRatio | Pixel aspect ratio of the media. |

### Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SequenceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <track>
    <audio>false</audio>
    <segment>
      <externalVideoMedia>
        <uri>http://www.example.com/my_video</uri>
        <format>mov</format>
        <essenceStreamId>0</essenceStreamId>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
```

```
        </timeBase>
        <samples>450</samples>
        <width>720</width>
        <height>576</height>
        <pixelAspectRatio>
          <horizontal>1</horizontal>
          <vertical>1</vertical>
        </pixelAspectRatio>
      </externalVideoMedia>
      <sourceTrack>1</sourceTrack>
      <in>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </in>
      <out>
        <samples>25</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </out>
      <sourceIn>
        <samples>0</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceIn>
      <sourceOut>
        <samples>25</samples>
        <timeBase>
          <numerator>1</numerator>
          <denominator>25</denominator>
        </timeBase>
      </sourceOut>
    </segment>
  </track>
</SequenceDocument>
```

CHAPTER

TEN

# USERS, GROUPS, AND ACCESS CONTROL

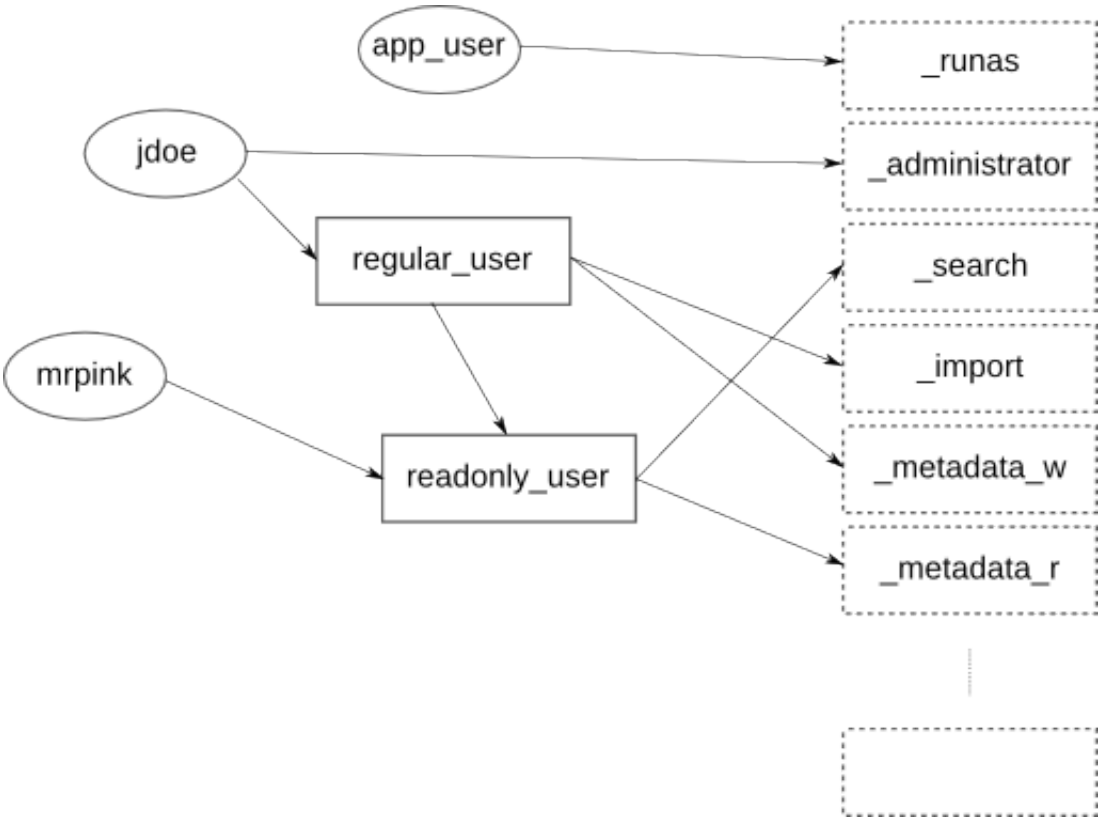
The user management system in Vidispine consists of users, groups, and roles.

- Roles are special groups, which cannot be added or deleted via the API.
- Regular groups and users can be added or deleted via the API.
- Users can belong to any number of groups or roles.
- Groups can depend on any number of groups or roles, although cyclic dependencies are not allowed.
- Roles cannot depend on any group or role.

To manage users and groups, see the *Users* and *Groups and roles* sections in the API reference.

## 10.1 Example

The following figure illustrates how users, groups and roles relate.



In the figure above, there are:

- Six roles: `_run_as`, `_administrator`, `_search`, `_import`, `_metadata_w`, and `metadata_r`.
- Two regular groups: `regular_user` and `readonly_user`.

  The group `readonly_user` depends on the roles `_search` and `_metadata_r`. The second group, `regular_user` depends on the roles `_import` and `_metadata_w`, and also the group `readonly_user`.

  In the last relation, `readonly_user` is called the **parent** group and `regular_user` is the **child** group. A user which belong to `regular_user` actually has all four roles.
- Three users: `app_user`, `jdoe`, and `mrpink`.

  The user `app_user` has the role `_run_as`, `jdoe` has the roles `_administrator`, `_search`, `_import`, `_metadata_w` and `_metadata_r` and `mrpink` has the roles `_search` and `_metadata_r`.

To visualize the users and groups like above, see *User/group visualization*.

## 10.2 Access control for items, libraries, collections

Items, libraries and collections have access control lists that determine what operations a user can perform. The entries in the list either corresponds to a specific user or to an entire group.

### 10.2.1 Overview

Vidispine uses the access controls on the item, library or collection to determine if a user has access to perform a specific operation or not.

All entities will have a OWNER access control that identifies the user that created the entity, and that grants full access to it. Below is an example access control list document showing the access that has been applied to a specific collection:

```
<AccessControlListDocument xmlns:ns0="http://xml.vidispine.com/schema/vidispine">
    <access id="VX-16610">
        <loc>http://vs.example.com:8080/API/collection/VX-16/access/VX-21</loc>
        <appliesTo>all</appliesTo>
        <permission>OWNER</permission>
        <user>admin</user>
    </access>
    <access id="VX-18037">
        <loc>http://vs.example.com:8080/API/collection/VX-16/access/VX-21</loc>
        <grantor>admin</grantor>
        <appliesTo>self</appliesTo>
        <appliesTo recursive="true">collection</appliesTo>
        <appliesTo recursive="true">item</appliesTo>
        <permission>READ</permission>
        <user>example-user</user>
    </access>
</AccessControlListDocument>
```

The first access entry is the OWNER access, which shows that the admin user has created the collection, and is thus the owner with full access.

The second access entry shows that admin has granted `READ` access to `example-user`. The `appliesTo` setting has been used to determine which entities the access control extends to. In this case, access has been granted to the collection itself, child collections and items, but not libraries. If the `appliesTo` element is not set, access is granted to self and all decedent entities.

New in version 4.17.7.

All `appliesTo` settings have a property called `recursive` which is used to control the depth of the accesses granted. A recursive setting will dig through the entities entire relationship tree until it finds all entities for which the setting is applicable. If a setting is not recursive it will only look at the direct children of the entity on which it is set. If recursive is not explicitly set, the default is that the setting IS recursive. Consider the following example:

- Collection A contains Item A and Collection B.
- Collection B contains Item B.

`<appliesTo recursive="true">item</appliesTo>` will affect both Item A and Item B. `<appliesTo recursive="false">item</appliesTo>` will only affect Item A.

Setting recursive on `self` is a no-op. Setting recursive on libraries is also no different from setting recursive to `false` since libraries cannot contain further libraries.

Manage access controls using the *access control resource* on the entity in question. For example, to grant access to Users, but only allow them access to certain shapes:

```
POST /collection/VX-16/access
Content-Type: application/xml

<AccessControlDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <permission>READ</permission>
  <group>users</group>
</AccessControlDocument>
```

```
POST /collection/VX-16/access
Content-Type: application/xml

<AccessControlDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <permission>NONE</permission>
  <group>users</group>
  <operation>
    <shape>
      <tag>original</tag>
    </shape>
  </operation>
</AccessControlDocument>
```

To view the access controls that apply for an item, including any access controls inherited from parent collections or libraries, see *Viewing applied access controls*.

### 10.2.2 Access levels

The higher levels grants the permissions of the lower levels.

**NONE** Grants no permissions whatsoever.

**READ** Grants permission to read.

**WRITE** Grants permission to write.

**ALL** The highest level that grants permissions to perform operations such as item deletion.

**OWNER** A specific case of ALL that is given by the system. This level cannot be added or removed.

### 10.2.3 Priority

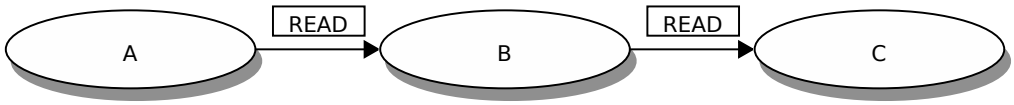
The access control lists are sorted in order to determine which entry that applies to a given operation. If there are multiple matching entries (i.e. match both the item and the operation being performed), following criteria are used to determine which entry applies, in order of most to least important:

1. **Explicit Priority** Controls with a high explicit priority take precedence over controls with lower explicit priority. An explicit priority can be assigned by setting the `priority` element in the AccessControlDocument to the desired level. The default is 0, and entries with a higher number override entries with lower value. Note that only superusers can create access controls with an explicit priority as users would otherwise be able to gain access to entities that they shouldn't have.
2. **Inheritance** Controls directly on the item take precedence over entries inherited from ancestor collections or libraries. It is not differentiated where the entry is inherited from, e.g. whether it is through several 'generations' of collections, or immediately from a library.
3. **User or Group** Entries granted directly to users take precedence over entries granted via groups.
4. **Operation Type** `Shape`, `Metadata`, and `Uri` entries take precedence over `Generic` entries. For `Metadata`, entries with a specific field set takes precedence over general metadata entries.
5. **Permission** Controls that grant more access take precedence over controls that give less access.

If no matching entry is found access will be denied.

### 10.2.4 Revoking access

The user that created an access control entry is also tracked. This is the *grantor*. It is also so that an entry is only valid if the grantor still has access to the entity. This means that access can be revoked by removing the original entry that granted access, or by disabling the grantor user without preserving access (see *Disable a user*).




For example, let's assume that user A is the owner and grants READ access to user B, that in turn grants READ access to user C, as shown in the figure. Users A, B and C now all have read access. If the access control granting READ access to user B then the user C will no longer have access.

### 10.2.5 Operation

There are different types of operations that can be restricted using access control lists. Parameters are optional and makes the access control entry more specific. If no operation is specified then the entry will be considered generic and apply to the entire item.

#### URIs

| Operation | `/item/` {item-id} `/uri` | |
|---|---|---|
| Parameters | type | The type of the URI to restrict. |

#### Example

```
<AccessControlDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <permission>READ</permission>
   <user>testuser</user>
   <operation>
      <uri>
        <type>lowres</type>
      </uri>
```

```
   </operation>
</AccessControlDocument>
```

### Shapes

| Operation | /item/ {item-id} /shape | |
|---|---|---|
| Parameters | tag | Restrict access to shapes with this tag. |

### Example

```
<AccessControlDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <permission>NONE</permission>
   <user>testuser</user>
   <operation>
      <shape>
        <tag>lowres</tag>
      </shape>
   </operation>
</AccessControlDocument>
```

### Metadata

| Operation | /item/ {item-id} /metadata | |
|---|---|---|
| Parameters | field | The name of the field to restrict. |

Changed in version 5.0: Comma separated field names are supported in <field>.

**Caution:** Removal of fields are currently not restricted

Currently fields can be removed without checking the specific access control entry.

### Example

```
<Accesscontroldocument Xmlns="http://xml.vidispine.com/schema/vidispine">
   <permission>READ</permission>
   <user>testuser</user>
   <operation>
      <metadata>
        <field>title</field>
      </metadata>
   </operation>
</AccessControlDocument>
```

## 10.3 Access control for metadata fields

Metadata field access control lists can be used to control the usage of metadata fields and metadata field groups at a global level, i.e. they apply to all items. The default behavior for a field or a group without any access control list is to grant everyone full permissions.

In case of a conflict, i.e. one or more entries in the access control list for a certain field or group applies to the same user - the entry granting the highest level of permissions apply.

Note that metadata field access control lists are applied after any other access control list have been applied. So for example a metadata field access control list won't grant a user access to a certain field of an item's metadata if the user cannot access the item in the first place.

### 10.3.1 Permission levels

There are four levels of permission, higher levels of permissions include all other permissions. The semantics of each permission differs depending on if it is associated with a group or a field.

| Permission | Field | Group |
|---|---|---|
| **NONE** | Grants no permissions whatsoever. | Grants no permissions whatsoever. |
| **READ** | Determines if user can see the contents of a field. | Allows for the group to be retrieved and seen when it is listed. Also allows for the group to be associated with items. |
| **WRITE** | Allows a user to set the value of a field. | Allows fields to be added and removed from the group. |
| **DELETE** | Allows a user to delete a field from the metadata of an item. | Allows deletion of the group. |

## 10.4 User authentication

Authentication of users in Vidispine can be performed in a number of ways depending on the requirements of the calling application.

1. By passing the user credentials to Vidispine on each request and letting Vidispine authenticate the user based on the credentials stored in the Vidispine database.

   The default HTTP authentication method is HTTP basic authentication. To use a custom HTTP authentication method, have a look at *Apache Shiro Integration*.

2. Using Run-As: The application can itself authenticate the user and then connect to Vidispine using a service account with the Run-As privilege and with the Run-As option enabled, so that the request is then performed as the already authenticated user.

3. Creating a time-limited token using the API with one of the options above, see *Retrieve an authentication token*. This token can then be used in subsequent calls as credential by specifying the HTTP header:

   ```
   Authorization: token {token}
   ```

4. Using long-lived *access keys*. Access keys are used with HTTP basic authentication, just like with normal username and password credentials.

### 10.4.1 Run-As option

The API supports the operation of having the calling application authenticate itself via a single password or a single certificate credential. The actual end-user can then be specified by the `RunAs` HTTP header. The calling application credential must have `_administrator` or `_runas` role. The actual end-user roles will be determined by the `RunAs` user's credentials.

A typical UI application scenario would be:

1. Have the user log in by providing user name and password.
2. Authenticate the user with *`PUT /user/(username)/validate`*.
3. Store the user name with the session.

4. Use the `RunAs` header with all communication to the Vidispine API.

### 10.4.2 Token authentication

The above scenario can also be achieved using short-lived authentication tokens.

1. Have the user log in by providing user name and password.
2. Request an authentication token using *GET /token* or *GET /user/(username)/token*.
3. Store the authentication token with the session.
4. Use the token to authenticate all communication to the Vidispine API.

The *GET /token* endpoint can be used when using access keys to authenticate, in case the username of the user is unknown.

### 10.4.3 Use access keys

Access keys can be seen as longed-lived authentication tokens, except that they do not expire. Multiple access keys can be created for a single user, and can be disabled so that they no longer can be used to successfully authenticate as that user. Deleting an access key will permanently disable it.

The only time the access key secret will be available is when the access key is first created.

To create an access key:

```
POST /user/stephen/key/
Accept: application/xml
```

```
<AccessKeyDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VSIDEWKIL4R27GPSJ562</id>
  <secret>pWUlPFHqyJom2Wq+XaGIUVRQqgx5jgnaLIY37/DT</secret>
  <status>ACTIVE</status>
  <created>2018-06-01T10:36:13.891+02:00</created>
</AccessKeyDocument>
```

The access key id and secret can the be used to authenticate further requests:

```
GET /whoami
Authorization: Basic␣
↪VlNJREVXS0lMNFIyN0dQU0o1NjI6cFdVbFBGSHF5Sm9tMldxK1hhR0lVVlJRcWd4NWpnbmFMSVkzNy9EVA==
```

```
200 OK

stephen
```

### 10.4.4 Apache Shiro Integration

As of Vidispine 4.1 requests can be forwarded to Apache Shiro (http://shiro.apache.org/) for authentication, making it is possible to customize how **existing** users in Vidispine are authenticated. The Apache Shiro version that is bundled with Vidispine can be seen in the table below.

| Vidispine version | Apache Shiro version |
|---|---|
| 4.12 | 1.4.0 |
| 4.1 | 1.2.2 |

### Custom configuration

On startup Vidispine will try to read a Apache Shiro INI configuration (http://shiro.apache.org/configuration.html#Configuration-INIConfiguration) file from `$instanceRoot/[config/]shiro.ini`. The instance root folder typically is `/var/lib/vidispine/server`. In case you are starting Vidispine manually via command line `shiro.ini` will be loaded from the current directory.

The default configuration file that can be used as a template can be seen below.

---

**Note:** The token authentication filter and the Vidispine realm *must* always be kept so that requests performed internally by Vidispine will still function.

---

```
[main]
vidispineRealm = com.vidispine.security.auth.DefaultVidispineRealm
tokenAuth = com.vidispine.security.auth.TokenAuthenticationFilter
deny = com.vidispine.security.auth.DenyFilter

securityManager.realms = $vidispineRealm
authcBasic.applicationName = vidispineRealm

[urls]
/** = noSessionCreation, tokenAuth[permissive], authcBasic
```

### Installing a custom filter or realm

1. Make the JAR file containing your custom filter or realm available on the classpath. With the `vidispine-server` package, the JAR file should be copied to `/usr/share/vidispine/server/lib/ext/`.
2. Create a `shiro.ini` file based on the above template and modify it to your needs.
3. Start/Restart the Vidispine service.

### Example: Static credentials

This is an example showing how to add a custom realm, in this case a IniRealm (http://shiro.apache.org/configuration.html#Configuration-%5Cusers%5C) that defines credentials for a static set of users directly in the configuration file.

```
[main]
vidispineRealm = com.vidispine.security.auth.DefaultVidispineRealm
tokenAuth = com.vidispine.security.auth.TokenAuthenticationFilter
deny = com.vidispine.security.auth.DenyFilter

securityManager.realms = $iniRealm, $vidispineRealm
authcBasic.applicationName = "vidispineRealm"

[urls]
/** = noSessionCreation, tokenAuth[permissive], authcBasic

[users]
admin=password
```

Testing the configuration:

```
GET /API/version HTTP/1.1
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
User-Agent: curl/7.32.0
Host: localhost:8080
Accept: */*

HTTP/1.1 200 OK
...
```

```
GET /API/version HTTP/1.1
Authorization: Basic YWRtaW46YWRtaW4=
User-Agent: curl/7.32.0
Host: localhost:8080
Accept: */*

HTTP/1.1 200 OK
...
```

```
GET /API/version HTTP/1.1
Authorization: Basic YWRtaW46aW52YWxpZA==
User-Agent: curl/7.32.0
Host: localhost:8080
Accept: */*

HTTP/1.1 401 Unauthorized
...
```

**Note:** By default Apache Shiro will accept a request if at least one realm accepts the provided credentials, which is why the passwords `password` (accepted by `iniRealm` and `admin` (accepted by `vidispineRealm`) are both accepted.

### Automatic creation of users

When using a custom Shiro realm to authenticate users, it may be the case that the user exists in the custom realm, but not in Vidispine. The Shiro authentication listener `com.vidispine.security.auth.UserCreationListener`, can be used to have users automatically created/updated after a successful authentication attempt, to match the user information provided by that realm.

In the `shiro.ini`, add the user creation listener:

```
[main]
...
userInfoProvider = com.example.CustomRealmUserInfoProvider
userCreationListener = com.vidispine.security.auth.UserCreationListener
userCreationListener.infoProvider = $userInfoProvider
userCreationListener.enableUserOnLogin = true
securityManager.authenticator.authenticationListeners = $userCreationListener
```

The `com.vidispine.security.auth.UserCreationListener` requires a `com.vidispine.security.auth.spi.VidispineUserInfoProvider` that can return the Vidispine user information from the user account information from Shiro/the custom realm.

The implementation of the `VidispineUserInfoProvider`, `com.example.CustomRealmUserInfoProvider` in the example above, should be placed in a JAR file available on the classpath, typically in the

/usr/share/vidispine/server/lib/ext/ directory.

Changed in version 4.17.4: The enableUserOnLogin field was added.

**Note:** OAuth2 and automatic user creation doesn't work together at the moment.

### OAuth 2.0

Version 4.4 contains a Shiro filter that can be used to authenticate Bearer tokens. To use, add the following to shiro.ini:

```
[main]
...
oauth2Auth = com.vidispine.security.auth.BearerAuthenticationFilter
...

[urls]
/** = noSessionCreation, tokenAuth[permissive], oauth2Auth[permissive], authcBasic
```

The validation of tokens can be done in three ways:

1. By checking the token against a plain public key or a public key in an X.509 certificate.
2. By checking the token against public keys given by federation metadata.
3. By checking the token against a validation provider.

### Example: static public key(s)

To set the static public key(s), add the following to shiro.ini:

```
[main]
...
oauth2Auth = com.vidispine.security.auth.BearerAuthenticationFilter
oauth2Auth.x509Certificate = {x509-certificateA}, {x509-certificateB}
oauth2Auth.publicKey = {publicKeyA}, {publicKeyB}
oauth2Auth.expectedAudience = {expected-audience}
oauth2Auth.tokenUser = email  # example
...

[urls]
/** = noSessionCreation, tokenAuth[permissive], oauth2Auth[permissive], authcBasic
```

Where {x509-certificate} is a X.509 certificate encoded with Base64, e.g. MII...== and, similarly, the {publicKey} is a Base64 encoded public key, e.g MII...AB. Token validation includes these steps:

- Vidispine circles through the certificates and public keys provided, e.g {x509-certificateA}, {x509-certificateB}`,``{publicKeyA}, {publicKeyB} and tries to verify the token's signature.
- The JWT claim sub must be present in the token (it's value isn't used).
- The JWT claim aud must contain an entry matching the {expected-audience} value.

After successful token validation Vidispine reads the JWT claim defined by the tokenUser property and uses it as Vidispine user name.

### Example: federation metadata

Federation metadata is similar to a static certificate, but multiple certificates can be used, and they are automatically downloaded regularly.

```
[main]
...
oauth2Auth = com.vidispine.security.auth.BearerAuthenticationFilter
oauth2Auth.federationMetadataURI = https://login.microsoftonline.com/common/
↪FederationMetadata/2007-06/FederationMetadata.xml
oauth2Auth.federationMetadataInterval = 86400
oauth2Auth.expectedAudience = https://graph.windows.net
oauth2Auth.tokenUser = unique_name
...

[urls]
/** = noSessionCreation, tokenAuth[permissive], oauth2Auth[permissive], authcBasic
```

### Example: validation service

Here, The token is validated against validation server. The result is stored in cache for 10 minutes.

```
[main]
...
oauth2Auth = com.vidispine.security.auth.BearerAuthenticationFilter
oauth2Auth.validationEndpoint = https://www.googleapis.com/userinfo/v2/me
oauth2Auth.tokenUser = email
...

[urls]
/** = noSessionCreation, tokenAuth[permissive], oauth2Auth[permissive], authcBasic
```

In an OpenID Connect (OIDC) context token validation usually will take place against the OIDC `userinfo` endpoint. Depending on your OIDC server this may require tokens with the `oidc` scope that are available in the OIDC hybrid flow.

In the example above with Google, the https://www.googleapis.com/auth/userinfo.email scope is used.

### Configure OAuth2 using the API

New in version 4.17.

You can update the OAuth2 configuration via the API. To enable/disable the API access, from the `shiro.ini` file, use the setting: `allowConfigUpdate` true/false. If there is no `shiro.ini` file present, the API access is enabled by default. Only users with the `_administrator` role can access the configuration from the API.

### Example

```
[main]
...
oauth2Auth = com.vidispine.security.auth.BearerAuthenticationFilter
oauth2Auth.x509Certificate = {x509-certificate}
oauth2Auth.expectedAudience = {expected-audience}
oauth2Auth.tokenUser = sub
oauth2Auth.allowConfigUpdate = false
...
```

```
[urls]
/** = noSessionCreation, tokenAuth[permissive], oauth2Auth[permissive], authcBasic
```

### Example

Get the current configuration.

```
GET /configuration/auth
Content-Type: application/xml

<OAuth2ConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <federationMetadataInterval>86400</federationMetadataInterval>
    <tokenUser>unique_name</tokenUser>
</OAuth2ConfigurationDocument>
```

### Example

Updating the configuration for a X.509 certificate.

```
PUT /configuration/auth
Content-Type: application/xml

<OAuth2ConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <x509Certificate>{x509certificate}</x509Certificate>
    <expectedAudience>{expected-audience}</expectedAudience>
    <tokenUser>email</tokenUser>
</OAuth2ConfigurationDocument>
```

### Example

Changed in version 5.6.

Updating the configuration for multiple X.509 certificates and public keys.

```
PUT /configuration/auth
Content-Type: application/xml

<OAuth2ConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <x509Certificate>{x509certificateA}</x509Certificate>
    <x509Certificate>{x509certificateB}</x509Certificate>
    <publicKey>{public-keyA}</publicKey>
    <publicKey>{public-keyB}</publicKey>
    <expectedAudience>{expected-audience}</expectedAudience>
    <tokenUser>sub</tokenUser>
</OAuth2ConfigurationDocument>
```

### Example

Updating the configuration for federation metadata.

```
PUT /configuration/auth
Content-Type: application/xml

<OAuth2ConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <federationMetadataURI>https://login.microsoftonline.com/common/
↪FederationMetadata/2007-06/FederationMetadata.xml</federationMetadataURI>
```

```
    <federationMetadataInterval>86400</federationMetadataInterval>
    <expectedAudience>https://graph.windows.net</expectedAudience>
    <tokenUser>unique_name</tokenUser>
</OAuth2ConfigurationDocument>
```

#### Example

Updating the configuration for validation service.

```
PUT /configuration/auth
Content-Type: application/xml

<OAuth2ConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <validationEndpoint>https://www.googleapis.com/userinfo/v2/me</validationEndpoint>
    <tokenUser>email</tokenUser>
</OAuth2ConfigurationDocument>
```

#### Example

Delete the current configuration.

```
DELETE /configuration/auth
Content-Type: application/xml
```

## 10.5 LDAP

Vidispine can authenticate users against an LDAP server and automatically synchronize users and groups from a directory at regular intervals if required.

### 10.5.1 User authentication

For users to be authenticated by an LDAP server, the server must first be configured in Vidispine.

1. An LDAP *resource* must be created, containing the connection details. There can currently only be one configured LDAP resource.
2. LDAP authentication must be enabled using the *ldapAuthentication* configuration property.

Users that are successfully authenticated will be added to Vidispine and will have the `_user` role by default.

#### Example: Enabling LDAP authentication

First, create the LDAP resource:

```
POST /resource
Content-Type: application/xml

<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <ldap>
    <url>ldap://someserver:389</url>
    <useStartTLS>false</useStartTLS>
    <userDN>cn=Users,dc=example,dc=com</userDN>
    <usernameAttribute>sAMAccountName</usernameAttribute>
    <userSearchFilter>(objectClass=user)</userSearchFilter>
    <bindDN>cn=Administrator,cn=Users,dc=example,dc=com</bindDN>
    <bindPassword>password</bindPassword>
```

```
  </ldap>
</ResourceDocument>
```

Then enable LDAP authentication:

```
PUT /configuration/properties
Content-Type: application/xml

<ConfigurationPropertyDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <key>ldapAuthentication</key>
  <value>true</value>
</ConfigurationPropertyDocument>
```

### Configuration

The elements in the LDAP resource are:

**url** The LDAP server(s) to connect to. Specify multiple servers to enable failover. Can be either `ldap://` or `ldaps://` (for SSL).

New in version 5.1.

To utilize the VSA *port forwarding service* feature for LDAP servers; the URL needs to be added as such: `vxa://<vxaUUID>:<id>`

---

**Note:** The scheme for the URL must be `vxa` and the port should refer to the ID of the *port forwaring service*. For example:

```
<url>vxa://e5817fdb-9deb-4f25-a689-72349a78407a:1</url>
```

---

**useStartTLS** Enables/disables StartTLS. Will be ignored when connecting using SSL.

**userDN** The user search base.

**userSearchFilter** The user search filter. The default is `(objectClass=*)`. The search filter and username attribute together define the filter that is used in the user query:

```
(@(`userSearchFilter`)(`usernameAttribute`=username))
```

If a single entry is found then a second bind is made to authenticate the user.

**usernameAttribute** The attribute that contains a users username/login name. Must uniquely identify a user. The default is `sAMAccountName`.

**realNameAttribute** The attribute that contains a users real name. The default is `cn`.

**cacheLifetime** Passwords are cached to reduce the number of requests made to the server. This element specifies how long password should be cached (in milliseconds). The default is `1800000` (30 minutes).

**usernameFormat** Can be set to `lower` to force Vidispine to lower case all usernames read from the LDAP server.

The bind properties can be set so that Vidispine authenticates using a bind request before searching for users or groups:

**bindDN** The DN of the entry to bind to before searching for a user.

**bindPassword** The password to provide in the bind request.

### 10.5.2 User and group synchronization

Vidispine can automatically synchronize users and groups, as well as user and group dependencies. Synchronization will be enabled if the `sync` element has been set.

Users from the directory that do not exist in Vidispine can be automatically created. If this should be enabled or not is typically a matter of:

- Licensing. If you are restricted to a certain number of users, then you may not want to create them in Vidispine if they are not using the system.
- Application needs. Access to an item can only be granted to a user that exists in Vidispine for example.

**Caution:** Password validation using *`PUT /user/(username)/validate`* will not work for imported users unless `type=raw`. This because a users password won't be available until the user has authenticated successfully at least once before. Validation should instead be performed using normal HTTP authentication.

#### Configuration

The `sync` element in the LDAP resource controls the synchronization:

**`sync`** If set then users and groups will periodically be updated from the LDAP server.

**`sync/interval`** The interval in milliseconds between synchronization attempts. The default is `1800000` (30 minutes).

**`sync/importOrganizationalUnits`** Indicates whether or not organizational units should be created as groups in Vidispine. Only units having users or groups will be added (as well as the parent units to these.)

**`sync/createUsers`** If new users should automatically be created. If `false`, then existing users will be updated by new/unknown users will be ignored.

**`sync/createGroups`** If new groups should automatically be created. If `false`, then existing groups will be updated by new/unknown groups will be ignored.

Old installations may still use the `import` element.

**`import`** Deprecated since version 4.0: The `import` element was previously used to enable synchronization. Use `sync` with `createUsers=true` and `createGroups=true` instead.

How groups are synchronized can be configured using the elements below.

**`groupDN`** The group search base. The default is the same as `userDN`.

**`groupSearchFilter`** The group search filter. The default is `(objectClass=group)`.

**`groupnameAttribute`** The attribute that contains a groups name. The default is `name`.

Subgroups are supported, that is, if the LDAP group query returns two groups, A and B, and B is listed as a member of A, then B will be added as a subgroup of A in Vidispine.

#### TLS configuration

New in version 5.3.2.

Two new optional elements have been added to control TLS (SSL) connection to control `ldaps` connection.

**`secureProtocol`** Controls which protocol to use. By default the standard Java setting is used (normally TLSv1.0). Recommended value is `TLSv1.2`.

**`serverCertificate`** Used to validate the server. Should be in PEM format, lines delimited by new-line character. Multiple certificates can be added.

Example:

```
-----BEGIN CERTIFICATE-----
MIIDd...
bmtub...
-----END CERTIFICATE-----
```

**Note:** If no certificates are given, *all* certificates are trusted.

### Examples

Importing all users from the **Users** organizational unit from an Active Directory server:

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <ldap>
    <url>ldap://example.com:389</url>
    <userDN>cn=Users,dc=example,dc=com</userDN>
    <usernameAttribute>sAMAccountName</usernameAttribute>
    <userSearchFilter>(objectClass=user)</userSearchFilter>
    <bindDN>cn=Administrator,cn=Users,dc=example,dc=com</bindDN>
    <bindPassword>{password}</bindPassword>
  </ldap>
</ResourceDocument>
```

Importing only members of a certain group:

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <ldap>
    ...
    <userSearchFilter>(&amp;(objectClass=user)(memberOf=cn=mam,cn=Groups,dc=example,
↪dc=com))</userSearchFilter>
    <groupSearchFilter>(&amp;(objectClass=group)(memberOf=cn=mam,cn=Groups,dc=example,
↪dc=com))</groupSearchFilter>
    ...
  </ldap>
</ResourceDocument>
```

Importing no groups, but creating groups to mirror the organizational unit tree structure.

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <ldap>
    ...
    <groupSearchFilter>(&amp;(objectClass=group)(|))</groupSearchFilter>
    <import>
      <importOrganizationalUnits>true</true>
    </import>
  </ldap>
</ResourceDocument>
```

The user Joe (`cn=Joe,ou=Users,dn=example,dc=com`) would then be added to the **Users** group.

### Trigger LDAP synchronization

This resource can be used to force a synchronization of users and groups, for example to verify that it is working properly.

**POST /resource/** (*type*) **/**
*resource-id***/sync** Triggers a synchronization of users and groups.

If users and groups are already synchronizing, than this will have no effect.

**Parameters**

- **type** – Must be *ldap*.

For example:

```
HTTP POST /resource/ldap/VX-1/sync

200 OK
```

### 10.5.3 Troubleshooting

If you are having problems with the LDAP integration then the best place to start is to check the LDAP *self test*. The test will connect to the LDAP server and list the users and groups that are found using the current configuration.

```
GET /selftest/ldap
Content-Type: application/xml

<SelfTestDocument xmlns="http://xml.vidispine.com/schema/vidispine" name="ldap"␣
↪status="ok" took="1ms">
  <message>No LDAP resource has been defined</message>
</SelfTestDocument>
```

You can also use tools such as ldapsearch (http://www.openldap.org/software/man.cgi?query=ldapsearch) or `ldp.exe` to verify the configuration:

```
$ ldapsearch -h ad.example.com -D "CN=VS,OU=Users,DC=example,DC=com" -W -b "OU=Users,
↪DC=example,DC=com"
```

If the configuration is correct, but users are still not being authenticated properly, then set the following log levels, try to authenticate once more and then check the application server log file to see what is going on.

```
com.vidispine.security=FINEST
com.vidispine.authentication=FINEST
```

For example, this error would indicate that the `userDN` element is missing:

```
Caused by: com.sun.enterprise.security.auth.realm.BadRealmException: A search base DN␣
↪must be provided.
        at com.vidispine.security.auth.realm.MultiRealm.init(MultiRealm.java:89)
        at com.sun.enterprise.security.auth.realm.Realm.doInstantiate(Realm.java:233)
```

#### Users are not assigned to the correct groups

Users will only be added to LDAP groups that have a corresponding group in Vidispine. If LDAP import is enabled then groups will also be created. Verify that the name attribute of the group corresponds to the name of the group in Vidispine.

Note that if a group is removed from the directory then the users will still be a part of the group. This is because we currently do not track which groups are to be synchronized with the groups from the directory, except by name.

### Users can only log in by entering their upper case username

What you can do is set `usernameFormat` to `lower` in the LDAP resource. Vidispine will then lower case all usernames read from the LDAP directory. Your users can then login by entering their username in lower case, or in any letter case if your application is lower casing usernames.

### Disabled the user can still login

A user will be marked as disabled if:

- The user has been removed from the directory.
- If the user has been disabled (Active Directory only.)

If users should be disabled based on some other criteria then update the user search filter so that it excludes users accordingly. For example:

```
(&(objectClass=user)(!(userAccountControl:1.2.840.113556.1.4.803:=2)))
```

### It's still not working

Contact us directly and we will try to figure out what's going on.

CHAPTER

ELEVEN

# MULTI-SITE

## 11.1 Multi-site

Vidispine supports syncing between remote sites, this is handled via *site rules*. In order to start using the multi-site capabilities, the sites must first be set up so they know about each other.

### 11.1.1 Site names

Every site must have a name. Out of the box, a Vidispine instance will have the site name `VX`. The site name determines what prefix the ID:s in the system will get (e.g. VX-1234) The site name can be changed by setting the Java system property `com.vidispine.site`. It is important that every site in a multi site setup have different names.

### 11.1.2 Multi site setup

Before anything can be synced between sites, Vidispine must be told how to connect to the remote sites. This is done by adding a site definition for each remote site. How this is done in practice is described in the *reference section*.

It is important to note that all sites must know about *all* other sites in order for the syncing to work properly! It is also important that the clocks on the different servers are set correctly, since for some operations the timestamps of changes are important.

### 11.1.3 Site rules

To determine which entities to sync to remote sites, Vidispine uses site rules. Site rules can be defined for a number of different entity types, and the rules can also define what parts of the item should be synced.

Site rules can be set for individual entities or collectively for all entities of a specific type (e.g. you could set a site rule applying only to item VX-100, or a rule that applies to all items in the system).

Site rules can be added for the following entity types:

- **Items**
- **Collections** All child entities will also be synced.
- **Libraries** All child items will be synced.
- **Users** Will also sync any parent groups.
- **User groups** Any child groups and users will be synced with the group.

Depending on what entity type the rule is posted to, a number of different settings are available. For item, collection and library rules, the following settings are available:

- **metadata** Whether or not to sync metadata
- **access** Whether or not to sync ACLs for the entities. This also requires that the affected users and groups have been synced, otherwise this setting will have no effect.

- **shape** Determines whether or not to sync a shape containing this tag.
- **files** Whether or not to also sync files or just shape information.

User and group rules have no special settings.

Setting a site rule on an entity will cause it to be synced to the remote site specified in the rule, and any future changes will also be synced. A synced item will be synced *both ways*. So any changes made on the remote site will be synced back to the original site as well.

#### Example

The following XML would describe a site rule for an item to the site NY. Metadata is synced, ACLs are not synced, and any `web` and `editing` shapes will be synced along with the files:

```
<SiteRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <site>NY</site>
  <metadata>true</metadata>
  <access>false</access>
  <shape>web</shape>
  <shape>editing</editing>
  <files>true</files>
</SiteRuleDocument>
```

### 11.1.4 Conflicts

When having a synced item on several sites, there is always the possibility of metadata conflicts occurring. In the Vidispine multi-site setup, it is handled as "last edit wins". Meaning that the edit with the latest timestamp will win. This does not mean that the older change is lost however. A full history of all edits will still be available on all sites, and the old value can be manually brought back with a later edit.

CHAPTER

TWELVE

# MISCELLANEOUS TOPICS

## 12.1 Deletion lock

New in version 4.15.

Deletion lock is a mechanism to prevent entities from been moved or deleted. Entities that support deletion lock are collections, items and files.

**Note:** New in version 4.17.2.

Locked files can be moved by running an explicit *move job*. Thus, overriding the lock for that file.

### 12.1.1 Adding locks

Deletion locks can be added to collections, items and files. All locks must have an expiration time, and can have user defined *key-value metadata* that can be used to *filter locks* and to for example explain why a lock exists.

```
<DeletionLockDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <expiryTime>2019-10-09T18:49:41.650+02:00</expiryTime>
  <metadata>
    <field>
      <key>reason</key>
      <value>Locked for playout</value>
    </field>
  </metadata>
</DeletionLockDocument>
```

Deletion locks can also be used in *searches*, and support *notifications* so that you can easily take action once a lock has expired.

### 12.1.2 Lock expiration

All deletion locks must have an expiration time.

- Expired locks are treated as if they weren't present at all.
- Expired deletion locks on collections and items can be automatically removed if `autoRemoveExpiredDeletionLocks` is set to `true`.
- Expired file deletion locks need to be removed manually.

The `isExpired` attribute highlights locks that have expired:

```
<DeletionLockDocument xmlns="http://xml.vidispine.com/schema/vidispine" isExpired=
↪"true">
  <id>VX-1574</id>
  <user>admin</user>
  <expiryTime>2018-09-09T18:49:41.650+02:00</expiryTime>
  <modified>2018-10-12T14:27:03.175+02:00</modified>
  <entityType>Item</entityType>
  <entityId>VX-32140</entityId>>
</DeletionLockDocument>
```

### 12.1.3 Working with multiple locks

An entity can have multiple deletion locks, but only zero or one *effective* deletion lock.

- An effective lock is a lock that hasn't expired, and has the maximum expiration time among all the explicit locks and inherited locks.
- An entity with an effective deletion lock can be copied, but not moved or deleted; either by any explicit API request, or Vidispine internally (storage-rule for example).

In the examples below, you will see that effective locks are shown as `isEffective="true`.

### 12.1.4 Lock inheritance

Deletion locks will be automatically inherited from parent to child entities, i.e., from collection to sub-collections to items and then to files, with one exception:

- If there is any explicit lock set on a file, the file will not inherit any parent locks.

One use case of this behavior is that if the original asset needs to be kept for a longer period of time, but its copy can be removed earlier.

### 12.1.5 Transient metadata

An entity's effective lock id and expiration time are added to the entity's metadata as *transient fields*:

- `__deletion_lock_id` - The id of the effective lock on the entity (`string-exact`).
- `__deletion_lock_expiry` - The expiration time of the effective lock (`date`).

For example, an item with a deletion lock will have metadata:

```
GET /item/VX-132340/metadata
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <item id="VX-132340">
  <metadata>
    <revision>VX-706676,VX-706677,VX-706678,VX-706671,VX-706672</revision>
    <timespan start="-INF" end="+INF">
      ...
      <field>
        <name>__deletion_lock_id</name>
        <value>VX-1559</value>
      </field>
      <field>
        <name>__deletion_lock_expiry</name>
        <value>2018-10-13T07:49:46.637Z</value>
      </field>
    </timespan>
```

```
  </metadata>
  </item>
</MetadataListDocument>
```

The same applies for files:

```
GET /storage/file/VX-10725401/metadata
```

```
<SimpleMetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <key>__deletion_lock_expiry</key>
    <value>2019-10-09T16:49:41.65Z</value>
  </field>
  <field>
    <key>__deletion_lock_id</key>
    <value>VX-1561</value>
  </field>
</SimpleMetadataDocument>
```

### 12.1.6 Examples

#### Collection lock inheritance

Assuming one collection and one item:

- The item belongs to the collection.
- The collection has an explicit lock set.

```
POST /collection/VX-9129/deletion-lock
```

```
<DeletionLockDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <expiryTime>2019-10-09T18:49:41.650+02:00</expiryTime>
</DeletionLockDocument>
```

Both the collection and the item will get the deletion lock.

```
GET /collection/VX-9129/deletion-lock
```

```
<DeletionLockListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <lock isEffective="true">
  <id>VX-1410</id>
  <user>admin</user>
  <expiryTime>2019-10-09T18:49:41.650+02:00</expiryTime>
  <modified>2018-10-11T15:04:24.125+02:00</modified>
  <entityType>Collection</entityType>
  <entityId>VX-9129</entityId>
  <metadata/>
  </lock>
</DeletionLockListDocument>
```

```
GET /item/VX-132271/deletion-lock
```

```
<DeletionLockListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <lock isEffective="true" isInherited="true">
  <id>VX-1410</id>
```

```
  <user>admin</user>
  <expiryTime>2019-10-09T18:49:41.650+02:00</expiryTime>
  <modified>2018-10-11T15:04:24.125+02:00</modified>
  <entityType>Collection</entityType>
  <entityId>VX-9129</entityId>
  <metadata/>
  </lock>
</DeletionLockListDocument>
```

### Effective locks

Assuming one collection and one item:

- Each of them have one explicit lock set.

```
POST /collection/VX-9129/deletion-lock
```

```
<DeletionLockDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <expiryTime>2019-10-09T18:49:41.650+02:00</expiryTime>
</DeletionLockDocument>
```

```
POST /item/VX-132271/deletion-lock
```

```
<DeletionLockDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <expiryTime>2019-09-09T18:49:41.650+02:00</expiryTime>
</DeletionLockDocument>
```

The item will inherit the lock from the collection. The effective lock of the item is the inherited one, since it has longer expiration time.

```
GET /item/VX-132271/deletion-lock
```

```
<DeletionLockListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <lock>
    <id>VX-1411</id>
    <user>admin</user>
    <expiryTime>2019-09-09T18:49:41.650+02:00</expiryTime>
    <modified>2018-10-11T15:16:42.802+02:00</modified>
    <entityType>Item</entityType>
    <entityId>VX-132271</entityId>
    <metadata/>
    </lock>
  <lock isEffective="true" isInherited="true">
    <id>VX-1410</id>
    <user>admin</user>
    <expiryTime>2019-10-09T18:49:41.650+02:00</expiryTime>
    <modified>2018-10-11T15:04:24.125+02:00</modified>
    <entityType>Collection</entityType>
    <entityId>VX-9129</entityId>
    <metadata/>
  </lock>
</DeletionLockListDocument>
```

### Explicit file locks

Assuming one item with one file in it:

- Both the item and the file have explicit locks set.

```
POST /item/VX-132164/deletion-lock
```

```
<DeletionLockDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <expiryTime>2019-10-09T18:49:41.650+02:00</expiryTime>
</DeletionLockDocument>
```

```
POST /file/VX-10725401/deletion-lock
```

```
<DeletionLockDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <expiryTime>2017-09-09T18:49:41.650+02:00</expiryTime>
</DeletionLockDocument>
```

The file will not inherit any lock from the item, since it has an explicit lock. In this case, the file lock has expired, so the file can be removed.

```
GET /item/VX-132164/deletion-lock
```

```
<DeletionLockListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <lock isEffective="true">
    <id>VX-1412</id>
    <user>admin</user>
    <expiryTime>2019-09-09T18:49:41.650+02:00</expiryTime>
    <modified>2018-10-11T15:39:30.402+02:00</modified>
    <entityType>Item</entityType>
    <entityId>VX-132164</entityId>
    <metadata/>
  </lock>
</DeletionLockListDocument>
```

```
GET /storage/file/VX-10725401/deletion-lock
```

```
<DeletionLockListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <lock isExpired="true">
    <id>VX-1413</id>
    <user>admin</user>
    <expiryTime>2017-09-09T18:49:41.650+02:00</expiryTime>
    <modified>2018-10-11T15:40:30.483+02:00</modified>
    <entityType>File</entityType>
    <entityId>VX-10725401</entityId>
    <metadata/>
  </lock>
</DeletionLockListDocument>
```

### Search using deletion locks

The *deletion lock transient metadata fields* can be used in collection, item and file searches. For example, to find items that will expire in the coming week:

```
PUT /search
```

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine" version="1">
  <field>
    <name>__deletion_lock_expiry</name>
```

```
    <range>
      <value>NOW</value>
      <value>NOW+7DAYS</value>
    </range>
  </field>
</ItemSearchDocument>
```

CHAPTER

THIRTEEN

# MONITORING

To get better insight into the operations of jobs and services you can collect metrics and traces in your favorite monitoring service. Metrics are exposed using JMX and StatsD (https://github.com/etsy/statsd/).

Transcoders on the other hand only expose metrics using StatsD.

## 13.1 StatsD

By default metrics are *not* sent to a StatsD server. To enable it you have to update the metrics configuration. For example, to have metrics sent to a StatsD server on `localhost` listening on UDP port 8125, use:

```
PUT API/configuration/metrics
Content-Type: application/xml

<MetricsConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <statsd/>
</MetricsConfigurationDocument>
```

Metrics sent to StatsD are by default prefixed with `vs.`. To have metrics sent with the prefix `vs1.`, for example if you have multiple instances running:

```
PUT API/configuration/metrics
Content-Type: application/xml

<MetricsConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <statsd>
    <host>metrics.example.com</host>
    <port>6125</port>
    <prefix>vs1</prefix>
  </statsd>
</MetricsConfigurationDocument>
```

Here metrics are sent to an external StatsD server on the non-standard port 6125. Note that the `.` between the prefix and metric name is added automatically.

### 13.1.1 Filtering metrics

You can set inclusion and exclusion filters to restrict which metrics are sent to the StatsD server. The default is to include all and exclude none.

Inclusion/exclusion filters may have a leading or trailing wildcard. For example, to exclude all `storage.fs` metrics:

```
<MetricsConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <statsd>
```

```
    <exclude>storage.fs.*</exclude>
  </statsd>
</MetricsConfigurationDocument>
```

### 13.1.2 Tagged metrics

Some metrics are tagged with additional information. These are sent to StatsD in the format:

```
<metricname>:<value>|<type>|#<tag>+
```

A `job.step.execution.time` metric might for example be sent as:

```
vs.job.step.execution.time:123|ms|#type:placeholder-import,step:100,sync
```

If your StatsD server does not support such tags then they can be disabled by setting `tags` to false:

```
<MetricsConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <statsd>
    ...
    <tags>false</tags>
  </statsd>
</MetricsConfigurationDocument>
```

## 13.2 JMX

Each metric is exposed as an JMX MBean in the "metrics" domain. You can view the metrics using for example:

- A JMX client such as VisualVM (http://visualvm.java.net/) with the VisualVM-MBeans plugin, or JConsole.
- Programmatically using the Java JMX client interface.
- Over HTTP/JSON using a bridge such as Jolokia (http://www.jolokia.org/).

## 13.3 Metrics

Metrics are exposed as either meters, timers or gauges. The name of a metric is meant to be self-explanatory. Timers are suffixed with `time` and meters are named as past tense verbs, while gauges make up the rest.

The StatsD type used for each metric, and the statistics exposed over JXM for each type are:

| Type | StatsD type | MBean attributes |
|---|---|---|
| Meter | `c` | The count, mean and 1/5/15-minute rates. |
| Gauge | `g` | The value. |
| Timer | `ms` | The count, min/max/mean/stdev, rates and percentiles. |

### 13.3.1 Indexing

- Meters:
  - `reindex.{index}.started`
  - `reindex.{index}.finished`
  - `indexer.solr.request.failed`
  - `indexer.elasticsearch.request.failed`
- Timers:

  - `indexer.solr.update.time`
  - `indexer.solr.delete.time`
  - `indexer.solr.commit.time`
  - `indexer.elasticsearch.update.time`
  - `indexer.elasticsearch.delete.time`
  - `indexer.{index}.index.time`
    * With `index` being one of `item`/`collection`/`acl`/`file`.
  - `indexer.library.update.time`

    Time spend on updating auto-refreshing libraries in the system.

### 13.3.2 Job

- Meters:
  - `job.created`
  - `job.started`
  - `job.finished`
  - `job.failed`
  - `job.blocked`
- Gauges:
  - `job.total.{state}`
    * Where `state` is the name of a *job state*, lower cased and with _ replaced with -. For example `finished-warning`.
- Timers:
  - `job.{type}.step.{step}.{sync}.execution.time`
  - `job.step.execution.time`
    * Tagged with `type:{type}`, `step:{step}` and `sync/async`.

### 13.3.3 Solr

- Meters:
  - `solr.request.failed`
- Timers:
  - `solr.query.time`
  - `solr.update.time`
  - `solr.commit.soft.time`
  - `solr.commit.hard.time`
  - `solr.optimize.time`

### 13.3.4 Elasticsearch

- Meters:
  - `elasticsearch.request.failed`
- Timers:
  - `elasticsearch.query.time`
  - `elasticsearch.update.time`
  - `elasticsearch.delete.time`

### 13.3.5 Storage

- Meters:
  - `storage.online`
    - Tagged with `storage:{id}`.
  - `storage.offline`
    - Tagged with `storage:{id}`.
  - `storage.method.online`
    - Tagged with `storage:{id}`.
  - `storage.method.offline`
    - Tagged with `storage:{id}`.
  - `storage.file.found`
    - Tagged with `storage:{id}`.
  - `storage.file.changed`
    - Tagged with `storage:{id}`.
  - `storage.file.deleted`
    - Tagged with `storage:{id}`.
  - `storage.file.hashed`
  - `storage.file.checksum.bytes.read`
  - `storage.fs.stat`
    - The number of `stat` call made.
- Gauges:
  - `storage.total.online`
  - `storage.total.offline`
  - `storage.total.evacuating`
  - `storage.total.evacuated`
    - The total number of storages with a specific state.

### 13.3.6 Resource

- Meters:
  - `resource.{type}.online`
    - Tagged with `resource:{id}`.
  - `resource.{type}.offline`
    - Tagged with `resource:{id}`.

### 13.3.7 Agent

- Gauges:
  - `agent.total.online`
  - `agent.total.offline`
    - The total number of agents with a specific state.

### 13.3.8 Transfer

- Meters:
  - `transfer.bytes.transferred`
  - `transfer.started`
  - `transfer.finished`
  - `transfer.finished-part`
  - `transfer.failed`
  - `transfer.blocked`

### 13.3.9 Service

- Meters:
  - `service.exception`
- Gauges:
  - `service.load.5`
    - The 5 minute load.
  - `service.load.60`
    - The 60 minute load.

### 13.3.10 Transcoder

- Gauges
  - `transcoder.{transcoder-id}.jobs.running`
  - `transcoder.{transcoder-id}.jobs.finished`
  - `transcoder.{transcoder-id}.jobs.failed`
  - `transcoder.{transcoder-id}.jobs.{transcoder-job-type}.running`
  - `transcoder.{transcoder-id}.jobs.{transcoder-job-type}.finished`

  - `transcoder.{transcoder-id}.jobs.{transcoder-job-type}.failed`

- Counters
  - `transcoder.{transcoder-id}.muxer.video.frames`
  - `transcoder.{transcoder-id}.encoder.{codec}.frames`
  - `transcoder.{transcoder-id}.decoder.{codec}.frames`
  - `transcoder.{transcoder-id}.io.{protocol}.{direction}.bytes`

### 13.3.11 Broker

- Gauges
  - `broker.queue.{queue}.size`

    The size of a specific queue. Note that this metric is only present when using the embedded broker.

### 13.3.12 Cluster

- Gauges
  - `cluster.size`

    The number of members in the cluster.

## 13.4 APM

Vidispine supports application performance monitoring using Elastic APM (https://www.elastic.co/products/apm). It monitors the execution of the application for easy pinpointing of performance issues.

### 13.4.1 Setup

In order to use Elastic APM you first need to set up an APM server (https://www.elastic.co/guide/en/apm/server/current/getting-started-apm-server.html). The elastic APM integration is disabled by default but can be enabled by adding the following configuration to the `server.yaml` file:

```
apm:
  elastic:
    urls: ["https://localhost:1234/"]
    secretToken: secret
    serviceName: vidispine
    serviceVersion: 5.0
    environment: staging
    sampleRate: 1
```

**Note:** The server will need to restart for any changes to take effect.

Please see the *APM configuration reference* for details.

Each trace encapsulates an event and may have one of the following types:

- `request`
  - A HTTP request, either incoming or outgoing.
- `messaging`

  - A JMS message, either incoming or outgoing.
- `scheduled`
  - A single iteration of a scheduled worker.
- `service`
  - A cross-object method invocation of a service layer class.
- `DB`
  - A database query

CHAPTER

FOURTEEN

# CONFIGURATION AND INTEGRATION

## 14.1 Search backend

Searching in Vidispine is implemented using either Solr or Elasticsearch as the backend. This allows functionality such as boolean operators, faceted searching, term highlighting, search term suggestions, etc.

### 14.1.1 Solr

Solr is the default search backend. It supports all features and works with both standalone Solr and SolrCloud.

### 14.1.2 Elasticsearch

#### Requirements

- Elasticsearch 6.8.1
- The analysis-icu (https://www.elastic.co/guide/en/elasticsearch/plugins/5.3/analysis-icu.html) plugin needs to be installed. If you are using Amazon Elasticsearch Service (https://aws.amazon.com/elasticsearch-service/), this plugin is already installed.
- Make sure to configure the *search backend* and that either the search backend URL or the `elasticsearchPath` property is set properly.
- Run `vidispine elasticsearch init` to initialize Elasticsearch. If an Elasticsearch instance is initialized already, the initialization will not create a new index, unless the `--recreate` argument is used when initializing.
- (Optional) Run `vidispine elasticsearch init --percolator-index` to initialize the "percolate query" index that is newly introduced in 5.6. This is useful for systems upgrading from an older version, and don't want to re-initialize everything. (New in 5.6.)
- Running `vidispine elasticsearch check` can be used to check if Elasticsearch is active already or should be initialized.

#### Limitations

The search interface is the same compared to using Solr (both syntactically and semantically), except that:

- *Collection-item join* is not yet supported.
- The `noescape` option in the "version 1" search document is not supported.
- When searching using cursor for timed intervals, the same item or collection might be returned with different timespans on different result pages. This is because timespans are sorted independently, but grouped to items on a per-result-page basis.

## 14.2 System configuration

### 14.2.1 Indexing configuration

The indexing configuration contains the parameters that relate to search and indexing.

- Where Vidispine can reach Solr or ZooKeeper.
- When to commit or soft commit.
- The Solr query request parameters.
- The default field settings.

This configuration replaces the configuration properties listed under *Search and indexing*.

#### Example

Full text indexing could be disabled for all fields, unless explicitly specified for a field, using:

```
<IndexingConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <solrPath>http://localhost:8088/solr</solrPath>
  <fieldDefault>
    <name>*</name>
    <fullText>false</fullText>
  </fieldDefault>
</IndexingConfigurationDocument>
```

### 14.2.2 Metrics configuration

See *StatsD* on how to configure how metrics are sent to StatsD. The configuration resource is described at *Metrics settings*.

### 14.2.3 FTP pool configuration

By default jobs that need to read or write to an FTP server will establish, use and end separate connections to the server. By configuring a FTP connection pool you can change so that the jobs share and reuse FTP connections. This can reduce the time it takes to transfer files over high latency connections.

For example, to create a connection pool with the default settings:

```
PUT /configuration/ftp-pool
Content-Type: application/xml

<FtpPoolConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <pool/>
<FtpPoolConfigurationDocument/>
```

If no pool is specified then pooling will be disabled. Unless overridden, the pool will be unbounded, and connections will expire after 1 minute. That is, the above configuration is identical to:

```
PUT /configuration/ftp-pool
Content-Type: application/xml

<FtpPoolConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <pool>
    <minSize>0</minSize>
    <maxSize>-1</maxSize>
    <evictionInterval>30000</evictionInterval>
```

```
    <minIdleTime>60000</minIdleTime>
  </pool>
</FtpPoolConfigurationDocument>
```

The FTP pool configuration resource is described at *FTP pool configuration*.

### 14.2.4 Database purging

Vidispine supports mechanisms for purging old information in database tables. Especially four tables can grow quite large without purging enabled.

#### Updated method for purging configuration

The preferred way of updating the database purging configuration is by create a DatabasePurgingConfigurationDocument, see *Database purging configuration*.

#### Change-log table

The change-log table holds information about data that should be sent to other *sites*. If multi-site is disabled (`disableSiteCrunching`), this table grows forever.

To enable purging of the table, two configuration properties are used: `changeLogPurgingTime` and `changeLogForcePurgingTime`. The first one controls deletion of entries that have been processed, the other one controls deletion of entries regardless of state.

Sensible values are 43200 and 86400, corresponding to one and two months, respectively.

Configuration can also be controlled via DatabasePurgingConfigurationDocument. For example:

```
<DatabasePurgingConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine
↪">
  <changeLog>
    <age>43200</age>
    <forceAge>86400</forceAge>
  </changeLog>
</DatabasePurgingConfigurationDocument>
```

#### Audit trail table

The audit trail table contains all API requests, see *Audit trails*.

To enable purging of the table, two configuration properties are used: `auditTrailPurgingTime` and `auditTrailPurgingDirectory`. Both must be set in order for purging to take place.

When purging is enabled, entries that are older than `auditTrailPurgingTime` minutes will be removed and put in a file inside the `auditTrailPurgingDirectory` folder.

A sensible value is 43200 or higher, corresponding to one month.

Configuration can also be controlled via DatabasePurgingConfigurationDocument. For example:

```
<DatabasePurgingConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine
↪">
  <auditTrail>
    <age>43200</age>
    <uri>s3://key:secret@archival-bucket/requests/</uri>
  </auditTrail>
</DatabasePurgingConfigurationDocument>
```

By setting the `compress` element in the configuration document, entries will be stored using gzip compression. The default value is true.

The entries are stored in batches. The number of entries per batch is controlled by the `batch` element in the configuration document. The default value is 10000 entries. If there are less than 10000 entries that fulfill the time criteria, the purging of the audit trail table will pause.

By setting the `body` element in the configuration document, entries will also include the request bodies and response codes. The default value is false.

The directory can be a full URI, such as a S3 or FTP location.

### Job table

To enable purging of the table, two configuration properties are used: *jobPurgingTime* and *jobPurgingDirectory*. Both must be set in order for purging to take place.

When purging is enabled, entries that are older than `jobPurgingTime` minutes will be removed and put in a file inside the `jobPurgingDirectory` folder.

Configuration can also be controlled via DatabasePurgingConfigurationDocument. For example:

```
<DatabasePurgingConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine
↪">
  <job>
    <age>43200</age>
    <uri>s3://key:secret@archival-bucket/job/</uri>
  </job>
</DatabasePurgingConfigurationDocument>
```

By setting the `compress` element in the configuration document, entries will be stored using gzip compression. The default value is true.

The directory can be a full URI, such as a S3 or FTP location.

### Transfer log table

Configuration is controlled via DatabasePurgingConfigurationDocument. For example:

```
<DatabasePurgingConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine
↪">
  <transferLog>
    <age>1440</age>
    <uri>s3://key:secret@archival-bucket/transfers/</uri>
  </transferLog>
</DatabasePurgingConfigurationDocument>
```

Both the `age` and `uri` element has to be set. Entries than represents a finished transfer older than `age` minutes are exported to the destination.

The `forceAge` element controls exports of non-finished transfers - transfers that have disappeared for some reason. The default value of `forceAge` is the value of the `age` element.

By setting the `compress` element in the configuration document, entries will be stored using gzip compression. The default value is true.

The entries are stored in batches. The number of entries per batch is controlled by the `batch` element in the configuration document. The default value is 10000 entries. If there are less than 10000 entries that fulfill the time criteria, the purging of the transfer log table will pause.

### 14.2.5 Default job priority

New in version 5.2.1.

The default job priorities are configurable by type. For example, the following configuration document will make `IMPORT` jobs default to `MEDIUM` priority and `EXPORT` jobs default to `HIGH` priority:

```
<JobPriorityConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <job type="IMPORT">MEDIUM</job>
  <job type="EXPORT">HIGH</job>
</JobPriorityConfigurationDocument>
```

If no default priority has been specified for a given job type, the job will default to `MEDIUM` priority.

### 14.2.6 CORS configuration

New in version 4.15.

Vidispine can be configured to emit Cross-Origin Resource Sharing (https://www.w3.org/TR/cors/) (CORS (https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS)) headers.

The CORS configuration is set using the *CORS configuration resource*. The configuration document consists of a number of entries, each of them are checked for CORS evaluation. If an entry condition matches the incoming request, the CORS headers set in the entry are outputted, and no other entries are matched. For example:

```
PUT /configuration/cors
```

```
<CORSConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <entry>
    <request/>
    <response>
      <allowOrigin>*</allowOrigin>
    </response>
  </entry>
</CORSConfigurationDocument>
```

The conditions in each entry can match HTTP method, the request path, the CORS origin, or any other header. For example:

```
<CORSConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <entry>
    <request>
      <pathRegex>API.*/item/.*</pathRegex>
      <headerRegex>
        <key>connection</key>
        <value>.*aliv.*</value>
      </headerRegex>
    </request>
    <response>
      <allowOrigin>*</allowOrigin>
      <allowMaxAge>86400</allowMaxAge>
    </response>
  </entry>
</CORSConfigurationDocument>
```

Semantics are as follows.

- Multiple methods can be set per entry. If one entry method matches the request method, the entry matches as far as for method. If no methods are set on the entry, the entry matches.

- Origins can be specified both exact or via regular expressions. If one entry origin matches the request origin, the entry matches as far as for origin. If no origins are set on the entry, the entry matches.
- Request paths can be specified via regular expressions. If one entry path matches the request path, the entry matches as far as for path. If no paths are set on the entry, the entry matches.
- Request headers can be specified via regular expressions. All specified headers for the entry must match the request. If the HTTP request does not contain a value for a specified header, the entry does not match. If the HTTP request contains multiple values for a specified header, it is sufficient that only one value matched the entry header condition.

When the entry matches, the entry may contain which origin that is allowed, the max age for the access, which methods that are allowed, which HTTP headers. The entry can also contain other, arbitrary, headers. If methods are not specified, the methods are automatically deduced from the API methods.

### 14.2.7 Configuration properties

Configuration properties are used in Vidispine to control system-wide parameters.

Since 5.3, configuration properties can be used to control some system property settings, see *below*.

**See also:**

See *Create/modify configuration properties* for more information about how to configure properties.

Some configuration properties are cached locally, and it may take up to 5 minutes until the new value is observed. These configuration properties are marked with "Cached: yes" below.

#### General

**apiUri**
URI to Application Server. Used by transcoder(s), so need to be a proper host if transcoder(s) run on another machine.

**Mandatory** Yes

**Example** `http://localhost:8080/API/`

**apiNoauthUri**
URI to Application Server, to use to access the no-auth API. Used by transcoder(s), so need to be a proper host if transcoder(s) run on another machine.

**Example** `http://localhost:8080`

**clusterName**
Optional alphanumerical identifier for the Vidispine installation/cluster. Must be set (to a unique identifier) if multiple Vidispine installations are to share a common set of transcoders.

**Example** `ABPRD`, `ABDEV`, `BCPRD`

**disableSiteCrunching**
Do not build site replication packages. Recommended to be set to `true` for systems not running site replication.

**Default** `true` since 4.2.6. Previous versions: `false`

**validatexml**
Enable schema validation of the incoming and outgoing xml document.

**Default** `false`

**slaveLicenseProxy**
Use a proxy for *Connection to Vidispine Online Licensing System*. Format is

•`http://IP:port/` or

- `socks://IP:port/`

Proxy authentication is not supported.

**Default** none

**defaultTranscoder**
Default transcoder resource to use. Valid values are:

- `vidinet` - To use the first available transcoder from Vidinet.
- A resource id - To use the transcoder with that id.

**Example** `VX-1`

**Default** none

### Search and indexing

Deprecated since version 4.2: The Solr and ZooKeeper properties are deprecated. Use *Indexing configuration* instead.

**solrPath**
URI ( **not path!** ) to Solr.

**Mandatory** Yes (No for SolrCloud)

**Example** `http://localhost:8081/solr/`

**elasticsearchPath**
URI to Elasticsearch's RESTful interface.

**Mandatory** Yes (if using Elasticsearch). Optional if `url` has already been configured in *search backend*

**Example** `http://localhost:9200/`

**zkHost**
For SolrCloud: A comma separated list of host:port pairs to the servers in the ZooKeeper ensemble.

**Mandatory** No (Yes for SolrCloud)

**Example** `localhost:3000,example.com:3001`

**solrCollection**
For SolrCloud: The collection in Solr to be used by Vidispine.

**Mandatory** No (Yes for SolrCloud)

**Example** `collection1`

**solrQueryTimeout**
The request timeout in milliseconds to use when querying Solr.

**Default** `60000`

**solrGroupLimit**
The maximum number of timespans to return per item or collection.

**Mandatory** No

**Example** `10`

**solrPingAttempts**
The number of times to ping a Solr node before aborting an active request.

**Default** `5`

**solrPingTimeout**
The request timeout in milliseconds to use when checking if a Solr node. is alive

**Default** `5000`

**solrCommitInterval**
The interval (in milliseconds) of Vidispine sending hard commit to Solr.

**Default** `10000`

**solrSoftCommitInterval**
The interval (in milliseconds) of Vidispine sending soft commit to Solr.

**Default** `-1` (disable)

**solrAutoSoftCommit**
If Vidispine should send soft commit to Solr automatically.

**Default** `true`

**solrUpdateQueueSize**
Number of documents Vidispine will send in batch to Solr.

**Default** `100`

**solrDeleteMergeSize**
The number of delete queries to merge into one before sending to Solr.

Changed in version 5.6: The default was changed from `-1` (do not merge any delete queries) to `100`.

**Default** `100`

**elasticsearchWorkerCount**
Number of worker threads to use when sending documents to Elasticsearch.

**Default** `1`

**Since** 5.6

**elasticsearchBulkBuffer**
The document buffer size (in bytes) in the Elasticsearch worker thread.

**Default** `10000`

**Since** 5.6

**indexFieldGroups**
If metadata field groups should be indexed in Solr. Setting this to `false` can reduce the load and the size of the index if items have a large number of groups in the metadata, but will mean that no results will be available when *searching for field groups*.

**Default** `true`

**indexCollectionItemOrder**
If the order of an item in a collection should be indexed in Solr. Settings this to `false` can greatly reduce the number of fields created in Solr and improve performance on systems with a lot of collections. This affects *collection item retrieval*. See also *Retrieve the child-collections of a collection*.

Recommended to be set to `false` for applications not relying on that feature. Requires a clean Solr index and a full re-index to take effect.

**Default** `false`

**indexTimespans**
If time coded metadata should be indexed in Solr. Setting this to `false` can reduce the load and the size of the

index if items/collections have a large number of timespans in the metadata, but will mean that no time coded metadata can be found.

**Default** `true`

**maxSearchResults**
Maximum number of search results allowed to be returned (see *Search items*).

**Default** `100`

**legacyTransientFieldTypes**
This setting controls the datatype of the transient metadata fields. If `true` then all transient fields will be of type `string`. If `false` the `*_size` and `*_count` fields will be of type `integer`, and the rest will have type `string`.

**Default** `true`

**skipLibraryIndexUpdates**
If set to `true`, the auto-refreshing libraries won't be updated after item metadata changes.

**Default** `false`

**indexDocumentMetadata**
If document metadata should be indexed. Setting this to `false` can reduce the load and size of the index. Document metadata is not *searchable* if this property is is set to `false`.

**Default** `false`

**Since** 5.0

**indexQueueLimit**
This setting throttles indexing to a maximum number of messages in the ActiveMQ queue. In a system where indexing (Solr/Elastic) is slower than the database, this setting avoids that too many messages are stored on the queue. Set this value to `0` to specify no limit. See also `activeMQAdminUrl`.

**Default** `0`

**Since** 5.7.1

**Cached** *yes*

### Metadata

**disableMetadataSchema**
If a metadata schema has been defined (see *Metadata schema*), allows metadata that does not comply to the schema.

**Default** `false`

**useAbsoluteSccTimeCode**
If set to true Vidicore will not try to adjust the time codes in the SCC sidecar file to the usual relative (zero-based) timecodes for each item when imported.

**Default** `false`

**Since** 5.6

### Bulky Metadata

**bulkyMetadataMigrationThreads**
Number of threads to use for bulky metadata migration.

**Default** `1`

**Since** 5.4.5

### Authentication

**passwordHashAlgorithm**
The hash algorithm used to hash all user passwords. Note that changing this will make it impossible to authenticate with any existing user.

**Default** `MD5`

**ldapAuthentication**
If set to true, *LDAP* authenticated will be enabled.

**Default** `false`

**userTokenMaxInterval**
Maximum token time for token created by regular user, in seconds.

**Default** `60`

**userTokenDefaultInterval**
Default token expiration time, in seconds.

**Default** `60`

**userTokenRefreshInterval**
Minimum time between token refreshments, in seconds.

**Default** `10`

### Jobs and imports

**concurrentJobs**
Number of *jobs* that are allowed to be started.

**Default** `3`

**dedicatedJobPool**
Enable/disable dedicated *job pool configuration*.

**Default** `false`

**jobRetryCount**
Number of retries for a job step before job continues with next step.

**Default** `5`

**jobExclusiveStepMaxWait**
The maximum number of seconds that a job step will wait before executing if there's a job step running from another job for the same item or file. This exists to reduce the number of optimistic locking exceptions for job steps that are known to conflict.

Only applies to steps with the exclusive flag (0x0100000) set.

**Default** `1`

**defaultIngestStorage**
The default destination storage for imports and transcodes. Note that storages selected by storage rules will take priority over this.

**Example** `VX-1`

**parseFileMetadata**
If set to true, file metadata will be metadata parsed and inserted as *Item metadata*. Supported formats for this type of metadata include Office formats and PDF files.

**Default** `false`

**maxFileMetadataLength**
The max length of file content (in characters) that will be extracted as *Item metadata*. `-1` means unlimited.

**Default** `100000`

**parseXMP**
If set to true, XMP metadata will be parsed and inserted as *Item metadata*.

**Default** `false`

**xmpIgnoreElements**
Contains comma-separated list of elements that are not read when parsing XMP data.

**Default** `DocumentAncestors,Pantry,History`

**simpleImageProcessor**
If false, use ImageMagick (must be installed, see Using ImageMagick for image handling (http://vidispine.tenderapp.com/kb/installation/using-imagemagick-for-image-handling)). Otherwise, use built-in image handling.

**Default** `true`

**disableThumbnailGeneration**
Will disable thumbnail generation by default. Can be overridden on a per job basis.

**Default** `false`

**alwaysGenerateThumbnails**
When `true`, thumbnails will be generated on import even if no transcoding takes place.

**Default** `false`

**disableThumbnailReindexing**
When `false`, the thumbnail index will be rebuild when items are *reindexed*.

**Default** `true`

**mediaCheckInterval**
The retry interval of media check (seconds).

**Default** `3`

### Storage and file

**groupImportableFiles**
When true, auto-import will only import one file for each file prefix as an item. Other non-sidecar files with the same prefix will be ignored. Set to false to import all files.

**Default** `true`

**keepMissingFiles**
If set to false then missing files that do *not* belong to any items will be removed from the database instead of being marked as lost.

Can be overridden on a per storage basis using the *keepMissingFiles* storage metadata property.

**Default** `false`

**keepEmptyDirectories**
Do not delete empty parent directories when deleting the last file in a directory, see *Parent directory management*.

Can be overridden on a per storage basis using the *keepEmptyDirectories* storage metadata property.

**Default** `false`

**scanMethodAlgorithm**
Set default scan method algorithm, see *Storage scanning algorithm*.

Can be overridden on a per storage basis using the `scanMethodAlgorithm` storage metadata property.

**Default** not set

**Since** 5.5.2

**storageActivationFile**
Require a .storage file to be present in the storage method's URI for storage to register as online.

**Default** `false`

**Since** 4.17

**fileHashAlgorithm**
Hashing algorithm used. If changed, the `c_hash` column of the `t_file` table should probably be set to `NULL`.

**Example** `SHA-1`

**enableTranscoderHash**
Off-load file hash calculation available transcoder.

**Default** `false`

**fileTempKeyDuration**
Number of minutes a no-auth URI is valid (*Auto method types*).

New in version 4.16: This property also controls the valid duration of a *VSA noauth URI*.

**Example** `10`

**useS3Proxy**
When `true`, Vidispine will create S3 pre-signed URLs for reading during job.

**Example** `false`

**s3ProxyValidTime**
The validate time (in minutes) of S3 pre-signed URL.

**Example** `60`

**s3ConcurrentParts**
The number of threads used for each S3 file upload.

**Default** `1`

**s3PartSize**
The S3 chunk/part size. Note that multipart uploads are always performed regardless of file size. Each part that is uploaded will be larger than the previous part. To control the increase of the part size, use `s3PartSizeIncrease`.

**Default** `5242880`

**s3PartSizeIncrease**
The size increase of each S3 part that is uploaded.

The default is chosen so that the maximum part size is 2 GB. With S3 supporting a maximum of 10000 parts, and the default part size being 5 MB, this gives a maximum object size of 3.5 TB.

**Default** Automatically selected based on the part size.

**s3ConnectionTimeout**
The timeout (in milliseconds) when establishing a connection to S3.

**Default** `50000`

**s3SocketTimeout**
The timeout (in milliseconds) when reading from a connection to S3.

**Default** `50000`

**s3MaxErrorRetry**
The maximum number of times to retry a failed S3 request.

**Default** `3`

**useAzureProxy**
When `true`, Vidispine will create AZURE-SAS URLs for reading during job.

**Example** `false`

**azureSasValidTime**
Specifies for how many minutes an AZURE-SAS URI will be valid. See *Retrieve a file*.

**Example** `60`

**stornextFileMetadata**
Specifies which fields that should be stored on the Vidispine file entity from StorNext metadata. See *StorNext Metadata*.

**Default** `location,class,existingCopies,targetCopies`

**useSegmentFiles**
If `true`, files generated by the transcoder on storages that do not support partial modification are written as segment files on the storage, instead of local files on the application server. See *Temporary storages for transcoder output*.

**Default** `false`

**useMutableRangeWrites**
If `true`, use a writing pattern that is more efficient for S3 writes. The only reason for not have this set to `true` is in a clustered set-up with transcoders directly connected to Vidispine Server, i.e., not via VSA.

**Default** `true`

**s3CredentialType**
Controls the type of S3 credentials being sent to a VSA. Allowed values are `none`, `temporary` and `secretkey`. For more explanation, please check *here*.

**Default** `temporary`

**stsCredentialDuration**
The duration (in seconds) of any temporary AWS credentials generated for agents. The allowed range of values is `[900,129600]`

**Default** `129600`

**stsRegion**
To generate temporary credentials Vidispine server will use the AWS Security Token Service (STS). Set this parameter to the region where you want Vidispine server to call the STS API. A good choice is the same region as your Vidispine server is running in.

**Default** `us-west-2`

**stsAssumeRole**
This is an optional configuration to use when generating credentials using the `s3` scheme. The name of the role to use when generating assume role credentials to give direct access to a file using *Generate temporary credentials*.

**Since** 4.15

**defaultStorageRuleJobPriority**
Controls which priority that should be assigned to jobs started as a result of storage rules, such as copy and delete jobs.

**Default** `MEDIUM`

**Since** 5.1

### Archival

**trustArchivedFiles**
A file needs to have a replica (another file with the same hash) before it can be removed by the storage rules.

If set to true, then archived files will be treated as valid replicas.

**Default** `false`

**glacierArchiveDescription**
Format the archive description according to the defined pattern:

- `{itemId}` - Replaced by the item id.
- `{fileId}` - Replaced by the id of the archived file.
- `{metadata-field:name}` - Replaced by the value of the metadata field with the given name.
- `{sourceId}` - Replaced by the id of the source file.
- `{sourcePath}` - Replaced by the path of the source file.
- `{sourceUri}` - Replaced by the URI of the source file.
- `{date}` - Replaced by the archive date in ISO 8601 format.
- `{dateString}` - Replaced by the archive date, in format `dow mon dd hh:mm:ss zzz yyyy`.

Example: `Item:{itemId},file:{fileId},path:{sourcePath},Archive date:{date},Title:{metadata-field:title}`

**Default** `my archive {dateString}`

### File system

---

**Tip:** Since 4.1.1, several of the `stat` system calls that was made by the JRE has been migrated into call in the JNI code. This can be enabled using the `localFSTimeData` option. On systems where local file systems are sensitive to stat loads, it is recommended to enable this option, and possibly the `statsPerSecond` option.

---

**fileHierarchy**
See *Using a tree structure for files*.

**Example** `0`

**fileSequenceStart**
The starting number for file sequences.

**Example** `0`, `0001`

**Default** `1`

**thumbnailHierarchy**
See *Using a tree structure for thumbnails*.

**Example** `0`

> **Warning:** Changing this property **will cause old thumbnails to be lost**. If you need to change the value on a system in production, please contact Vidispine.

**statsPerSecond**
Limit the total number of stats done on local file system. See also per-storage metadata ( *Storages*).

**localFSTimeData**
Use JNI methods for retrieving file modification time. See below.

**Default** `false`

**firstLastModifiedAsCreationTime**
Use the first reading of modification time as the creation time. Can be used on file systems which do not have the notion of creation time.

**Default** `false`

**disableATime**
Do not record atime. Used in conjunction with *`localFSTimeData`*.

**Default** `false`

### Transfers

**signiantManagerHost**
Hostname of Signiant manager. See *Signiant Integration*

**signiantManagerUser**
Username for Signiant manager. See *Signiant Integration*

**signiantManagerPassword**
Password for Signiant manager. See *Signiant Integration*

**enableTranscoderTransfer**
Off-load file-to-file transfers of non-growing files to available transcoder.

Changed in version 5.3.1: Default value was changed to true

**Default** `true`

**storageRuleDisableArchiveSources**
If `true`, archived files will not be used as sources for storage rule transfers.

**Default** `false`

### Library

**libraryUpdateInterval**
Default library update interval in the system (seconds).

**Default** `60`

**libraryExpireTime**
Default library expire time in the system (seconds).

**Default** `86400`

**useLucene**
If Lucene should be used directly when updating auto-refreshing libraries. This is faster than using Solr when there are a large amount of auto-refreshing libraries, but only works with the default Solr configuration that is shipped with Vidispine.

**Default** `false`

### Growing files

**`fileGrowingTimeout`**
The max time a file can keep growing (seconds).

**Default** `36000`

**`fileNotGrowingTimeout`**
A file is considered as not growing if it has not been changed during this period (seconds).

**Default** `600`

### JavaScript

**`javascriptInterpreter`**
The default *JavaScript engine* to use for scripts that don't explicitly target a specific engine. Valid values are:

- `graalvm` - Use GraalVM JavaScript.
- `rhino` - Use Mozilla Rhino.

Changed in version 5.0: GraalVM JavaScript was made the default.

**Default** `graalvm`

**Since** 5.0

### Services

**`itemDeleteInterval`**
The running interval (seconds) of `ItemDeleteCruncher` during the "idle" period (no item to delete).

**Default** `60`

**`itemDeleteIntervalShort`**
The running interval (seconds) of `ItemDeleteCruncher` during the "busy" period (there are items to be deleted).

**Default** `5`

**`itemDeleteExecutionTime`**
Max running time (seconds) of `ItemDeleteCruncher` thread, after that it goes to sleep.

**Default** `5`

**`fileHashExecutionTime`**
Max running time (seconds) of a file hashing thread, after that it goes to sleep.

**Default** `10`

### Broker

**`compressDocumentMessages`**
If JMS messages containing XML should be compressed or not. If `true` then the JMS_SUN_COMPRESS (http://docs.oracle.com/cd/E19798-01/821-1796/aeqdf/index.html) property will be set on JMS messages so that compression/decompression is performed by the OpenMQ client.

Works only with OpenMQ.

**Default** `true`

**activeMQAdminUrl**
Specifies the admin URI of ActiveMQ, for example `http://myactivemq:8761`. Normally it is not necessary specify it, but some functionality requires it, for example *indexQueueLimit*. For embedded broker, this does not have to be set.

**Default** not set

**Since** 5.7.1

**Cached** *yes*

**activeMQBrokerName**
Specifies the ActiveMQ broker name used. Used in conjunction with *activeMQAdminUrl*.

**Default** `localhost`

**Since** 5.7.1

**Cached** *yes*

### Database management

**auditTrailPurgingTime**
Remove all audit trail entries older than the specified time (in minutes) and put them in XML format in files inside the directory described by `auditTrailPurgingDirectory`. See *Audit trail table*.

**Default** not set

**auditTrailPurgingDirectory**

**Default** not set

**auditTrailPurgingCompress**
If `true`, purged logs are stored using gzip compression. If `false`, they are stored as plain XML documents. Equivalent to the `compress` element in the *database purging configuration document*

**Default** true

**auditTrailPurgingBatch**
Sets batch size of audit log entries when puring. If there are fewer entries, purging is paused. Equivalent to the `batch` element *database purging configuration document*

**Default** 10000

**auditTrailIncludeBody**
When `true`, the audit trail will include the body of requests, as well as the response code returned by the requests. These will always be included in purged documents, and can be seen in *GET /log* requests using the `body` query parameter. Equivalent to the `body` element in *database purging configuration document*

**Default** false

**Since** 5.1

**changeLogPurgingTime**
Remove all processed change-log entries older than the specified time (in minutes). See *Change-log table*.

**Default** not set

**changeLogForcePurgingTime**
Remove all change-log entries (processed or unprocessed) older than the specified time (in minutes).

**Default** not set

**jobPurgingTime**
Remove all job entries older than the specified time (in minutes) and put them in XML format in files inside the directory described by `jobPurgingDirectory`. See *Job table*.

**Default** not set

**jobPurgingDirectory**

**Default** not set

**disableSequenceChecker**

**Default** false

On start-up, Vidispine checks the sequences for all tables, which can be a lengthy process. On a database which is known to be in a consistent state, setting `disableSequenceChecker` to `true` will cause this step to be skipped.

### Transcoding

**maxTranscoderUnavailableTime**
When a transcoding job has started, and transcoder connection becomes available, wait for this time (seconds) for connection to be restored until job fails.

**Default** 60 seconds

**bulkyMetadataKeysToIgnore**
Comma-separated list of bulky metadata keys to ignore from analysis results, e.g. `crop`,

**Default** (none)

**transcoderNonblockingStatusInterval**
How frequently the transcode progress of a job will be updated, in milliseconds. A lower number may give a better user experience, but also a higher number of writes to the database.

**Default** 5000

### Vidispine Server Agent

**syncVxaFileChanges**
This determines whether items created from the VSA or VDA are re-created when the corresponding original file is changed. If the original file is deleted, the item would be removed.

**Default** false

**syncVxaDeletes**
Similar to `syncVxaFileChanges` but this one only affects file deletions. When the original file on an agent storage is removed, the corresponding item in Vidispine would also be deleted.

**Default** true

**useVxaHash**
If set to `true`, Vidispine agent will be used to compute the hash of the files.

**Default** false

**useVxaMimeType**
If set to `true`, Vidispine agent will be used to detect the mime type of the files.

**Default** false

#### Deletion lock

**autoRemoveExpiredDeletionLocks**
It set to `true`, expired deletion locks on collections and items will be removed automatically

**Since** 4.15

**Default** false

**deletionLockCleanUpBatchSzie**
Number of "to be removed" or "expired" deletion locks that will cleaned up in one batch.

**Since** 4.17.7

**Default** 100

### 14.2.8 System properties

System properties are set as argument to the JVM. See *Setting JVM options*.

The following properties are used in Vidispine:

**com.vidispine.site**
The site id prefix for the *current site*.

**Default** `VX`

**com.vidispine.license.dir**
The directory containing the Vidispine license or slave license file.

**Default** `${com.sun.aas.instanceRoot}`

**com.vidispine.license.tmpdir**
The directory where temporary license files may be stored.

**Default** `${com.sun.aas.instanceRoot}`

**com.vidispine.credentials.dir**
The directory containing credentials files such as the `AwsCredentials.properties` file used with Amazon S3 and Glacier.

**Default** `${com.sun.aas.instanceRoot}`

**com.vidispine.log.dir**
The directory containing the server log files.

**Default** `${com.sun.aas.instanceRoot}/logs`

**vidispine.identifier.format**
If `full`, output *Long identifiers*.

**Default** Normal, short identifiers.

**com.vidispine.xml.prefix**
Controls namespace prefix mapping of XML written by Vidispine that is _not_ part of API output, e.g. XML written to files. Comma-separated list of `{namespace}={prefix}` assignments. Namespaces without assignment get a `ns#` prefix.

**Default** Only `ns#` prefixes.

**Example** `http://www.smpte-ra.org/schemas/2067-2/2016=cc`

**infi.sleep_after_start**
The number of seconds to sleep after starting Infinispan, before VidiCore going on starting other services.

This should prevent "split-brain" issue during startup in some cases.

**Default** `-1` (No delay)

**`com.vidispine.asyncpool.coresize`**
Controls the size of the internal async pool in VidiCore (New in 5.3.1.). It's also available as a configuration property.

**Default** 5

**`com.vidispine.service.quorum`**
The VidiCore background services (i.e. `IndexCruncher`, `JobCruncher`, etc.) would only be started/stopped if the quorum is reached. This is to prevent "non-clustered" services from being started on multiple nodes in a cluster with a "split-brain" issue.

The quorum value should be set to `> N/2`, where `N` is the total instance count in the cluster.

For Example: The quorum should be set to `2` for a two-node-cluster as well as a three-node-cluster.

(New in 5.3.5.).

**Default** 1

Since 5.3, some system properties can be overridden by configuration properties. The properties that can be overridden are:

- `com.vidispine.site`
- `vidispine.identifier.format`
- `com.vidispine.xml.prefix`

Also in 5.3, is is possible to change the log levels of components using configuration properties. The configuration properties used for this is:

- `loglevel.` followed by the component name. (e.g. `loglevel.com.vidispine.filemgmt.storagesupervisor`)

New in version 5.4.4.

It is now also possible to override the `com.vidispine.service.quorum` system property using a configuration property as above.

While it is possible to change these values on a system in production, it is adviced that changing these values are done on an idle system, as the update in a cluster is asynchronous.

### 14.2.9 Bulky metadata storage

New in version 5.3.

By default, the values of *bulky metadata* are stored in the database. In a large system, this can occupy a large portion of the database. In version 5.3, there is the possibility to store bulky metadata on file system (or cloud storage).

To change the configuration the *bulky metadata configuration resource* is used. The configuration document contains two parameters, a base URI for where the bulky metadata should be stored, and an option to disable that storage. An explanation on how these to parameters interact follows below.

#### Transferring bulky metadata from database to file system

To move bulky metadata entries from database to the file system pointed out by the URI, use a configuration document like this:

```
PUT /configuration/bulkymetadata
Content-Type: application/xml

<BulkyMetadataConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
```

```
  <uri>file:///mnt/srv/bulkystorage/</uri>
</BulkyMetadataConfigurationDocument>
```

When this configuration is set, all bulky metadata is written to the file storage. Reading metadata is also done from file storage, but if the file storage does not contain the metadata, the database is read instead.

The path that is used to store the bulky metadata looks like this:

- Item id, with the id split in thousands
- Shape id, with the id split in thousands
- Key, channel, stream, and timecode, delimited by −.

In order to migrate all metadata to the storage, a reindex command is used:

```
PUT /reindex/bulkymetadata
```

When the reindex process has finished, all metadata is on the storage. This can be verified by retrieving the bulky metadata configuration, which also returns status information.

```
GET /configuration/bulkymetadata
Accept: application/xml

<BulkyMetadataConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>file:///mnt/srv/bulkystorage/</uri>
  <status>
    <metadataInDatabase>0</metadataInDatabase>
    <metadataOnStorage>45033</metadataOnStorage>
    <storageStatus>OK</storageStatus>
  </status>
</BulkyMetadataConfigurationDocument>
```

New in version 5.4.5: The bulky metadata migration can be parallelized by setting the *bulkyMetadataMigrationThreads* property.

### Transferring bulky metadata from file system to database

To move bulky metadata entries back from the file system to database, use a configuration document like this:

```
PUT /configuration/bulkymetadata
Content-Type: application/xml

<BulkyMetadataConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>file:///mnt/srv/bulkystorage/</uri>
  <storageDisabled>true</storageDisabled>
</BulkyMetadataConfigurationDocument>
```

Note the URI is still present, but the storageDisabled is set to true. It is vital that the URI remains set, and unchanged, as metadata will be read from the storage if it is not present in the database.

To migrate all metadata, use again:

```
PUT /reindex/bulkymetadata
```

```
GET /configuration/bulkymetadata
Accept: application/xml

<BulkyMetadataConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
```

```
  <uri>file:///mnt/srv/bulkystorage/</uri>
  <storageDisabled>true</storageDisabled>
  <status>
    <metadataInDatabase>45033</metadataInDatabase>
    <metadataOnStorage>0</metadataOnStorage>
    <storageStatus>OK</storageStatus>
  </status>
</BulkyMetadataConfigurationDocument>
```

Once everything has been migrated, the URI can be removed from the configuration, if preferred.

#### Transferring bulky metadata from one storage to another

There are two ways of moving bulky metadata from one storage to another.

1. Move metadata from storage to database, ensure every metadata is moved, then move metadata to new storage.
2. Stop/pause Vidispine, copy or move files from the old storage to the new storage. Start Vidispine and change URI.

If the second option is chosen, tools like rsync or aws s3 sync can be used to do a first synchronization while the system is in use.

#### Important notes

**Note:** The number of bulky metadata entries on storage returned by `GET /configuration/bulky-metadata` is based on database information, and not explicit file system scanning.

**Note:** When the bulky metadata is moved from database to file system, note that database backups no longer contain the bulky metadata values. Make sure appropriate backup procedures is in place for the bulky metadata storage. Good tools are rsync and aws s3 sync (mentioned above) or incremental tar balls.

**Note:** Before migrating bulky metadata, a full database backup is recommended.

### 14.2.10 Advanced configuration/tweaking

Methods in this section should only be used after recommendation by Vidispine Support.

#### SQL Query rewriting

New in version 5.0.6.

SQL queries sent by Vidispine can be modified by creating a file `sqltranslations.txt` in the `/etc/vidispine/` directory. The format of the file are pairs of lines, where the first line of each pair is a regular expression (https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html), and the second line of the pair is a replace pattern, e.g.:

```
(?i)^(select .*)$
/* comment inserted */ $1
(?i)^(delete .*)$
/* another comment inserted */ $1
```

## 14.3 External identifiers

External ids can be set on entities to provide an alternate way of accessing them. For example, instead of using the id VX-100 to access a particular storage, a custom external id such as `example_storage` can be used.

An external id is defined as the triple (entity type, namespace, value) and must be unique among the same entity type. The namespace is defined as the tuple (name, regular expression), where the regular expression is used to determine which namespace a given external id belongs to. It is therefore preferred that the set of possible values of a regular expression for a namespace is disjoint from the set of values for another namespace. Further note that set of values must also be disjoint from any *ids generated by the system*.

Managed namespaces using the *namespace resource*.

### 14.3.1 Priority

If the sets of possible values for all namespaces are disjoint, then no conflict exists. However, if they do have values in common there is an ambiguity regarding which namespace a particular value belongs to. This ambiguity is solved by that all namespaces have a priority value. Upon retrieving the namespaces they will be sorted in ascending ordering according to the priority value (thus making a namespace with a smaller value more important than a namespace with a larger value).

#### Example

Given two namespaces, N1, matching alphanumeric strings, and N2, matching all strings, with N1 being more important than N2, then N1 will always match alphanumeric strings and N2 will match all other strings.

| Namespace | Pattern | Priority |
|---|---|---|
| N1 | `[a-zA-Z0-9]+` | 5 |
| N2 | `.+` | 10 |

If the priority values were reversed so that N2 had the smaller priority value, it would match all strings and N1 would match no strings.

### 14.3.2 Example: The UUID namespace

In this example we will create a namespace for UUIDs and assign a UUID as an external identifier for an item.

First we create the namespace and simply name it “uuid”.

```
PUT /external-id/uuid
Content-Type: application/xml

<ExternalIdentifierNamespaceDocument xmlns="http://xml.vidispine.com/schema/vidispine
↪">
    <pattern>[A-Fa-f0-9]{8}\-[A-Fa-f0-9]{4}\-[A-Fa-f0-9]{4}\-[A-Fa-f0-9]{4}\-[A-Fa-f0-
↪9]{12}</pattern>
    <priority>10</priority>
</ExternalIdentifierNamespaceDocument>
```

Then we assign a UUID to the item VX-11.

```
PUT /item/VX-11/external-id/69e436fe-eaed-4061-a66b-7d7c4bf80b20
```

Retrieving the definition:

```
GET /item/69e436fe-eaed-4061-a66b-7d7c4bf80b20/external-id
```

```
<ExternalIdentifierListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>
    <entityId>VX-11</entityId>
    <entityType>Item</entityType>
    <namespace>uuid</namespace>
    <externalId>69e436fe-eaed-4061-a66b-7d7c4bf80b20</externalId>
  </id>
</ExternalIdentifierListDocument>
```

## 14.4 License handling

Vidispine requires a valid license in order to run. The license controls how many items and storages may exist in the system, and controls which encoders and decoders are available when transcoding.

License keys for on-premise systems can be obtained by contacting your local Sales Representative. When purchasing Vidispine through Vidinet, all license handling is fully automated.

### 14.4.1 How it works

The license is a physical file which must reside in the Vidispine root folder (by default `/etc/vidispine/`). Vidispine can read two different types of license files:

- `slaveAuth.lic` - will cause Vidispine to connect to the global Vidispine Online Licensing System to obtain its license properties. Your server must be able to reach the internet on port 8080 in order to become properly licensed.
- `License.lic`- (Deprecated since version 5.0) a fallback licensing method for use in airgapped solutions that locks the license to the systems MAC address. This method is used for the *Deployment License*.

Standalone Vidispine transcoder nodes are licensed through the API and do not need a local license file.

#### License types

**Development/test/demo license** This type of license allows for unlimited numbers of everything, for a certain period of time. All transcoded content will be watermarked.

**Production license** This type of license will allow for a certain number of users, assets, storages, transcoders and codecs according to what's purchased. Transcoded content will *not* be watermarked.

**Deployment license** When installing Vidispine using the installer, a non-MAC bound deployment license will be installed. This license will allow you to verify that your system was properly installed. The license will allow for 2 users, 100 assets, 1 transcoder, 1 storage area, and encoding/decoding of all codecs supported by Vidispine. All transcoded content will be watermarked.

#### License errors

If the license file is missing or if you have exceeded the license limits, a HTTP response 402 Payment Required (http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.3) will be returned. Details on what limit(s) have been exceeded can be found from *GET /version*.

The response will also display the version numbers for the various installed Vidispine components.

#### Connection to Vidispine Online Licensing System

The connection to the Vidispine Online Licensing System is done via HTTP and is tolerant to temporary error and outages.

If your infrastructure does not allow outbound HTTP connections, a proxy service can be used see *slaveLicenseProxy*.

### 14.4.2 Redundancy and timeouts

When a system is licensed using a `slaveAuth.lic` file, a heartbeat request will be sent every 60 seconds towards the Vidispine Online Licensing System, using the endpoints listed in the `slaveAuth.lic` file. The system will perform an automatic fail over to the second endpoint if the primary endpoint is offline or does not respond in a timely fashion. If the system does not receive a valid response from any endpoint, an error will be logged. If the system does not receive a valid response from any endpoint within 5 days, the system will enter read-only-mode, impairing normal usage. A request to *GET /version* on the system will report "License status invalid". The system will go back to normal operation as soon as a valid response from any license server endpoint is received.

## 14.5 Using JavaScript to extend operations

JavaScript can be used to add integration code in a number of places, such as job tasks, transcode presets, naming scripts, etc. This article describes functions and utilities that are common to all JavaScript invocations.

If a script is not working as expected, then it is also possible to *debug the script* using Eclipse.

### 14.5.1 JavaScript engines

There are two JavaScript engines that can be used for evaluating JavaScript.

**GraalVM JavaScript** GraalVM JavaScript (https://github.com/graalvm/graaljs) is an ECMAScript 2019 compliant JavaScript implementation built on GraalVM. This is the default engine.

**Rhino** Rhino (https://developer.mozilla.org/en-US/docs/Rhino) is the legacy engine that has been used since the Vidispine project was started. Rhino supports ECMAScript for XML (E4X) (https://en.wikipedia.org/wiki/ECMAScript_for_XML).

New in version 5.0: GraalJS was added as a script engine.

#### Selecting a JavaScript engine

The default JavaScript engine can be configured using the *javascriptInterpreter* configuration property, but the JavaScript engine to use can also be specified using a comment in the script itself.

The format is:

```
/* interpreter=graalvm|rhino */
```

For example, this script would be evaluated using GraalJS:

```
/* interpreter=graalvm */
let xs = [1, 2, 3];
xs = xs.map(x => x + 1);
xs
```

While this would use Rhino:

```
/* interpreter=rhino */
var xs = [1, 2, 3];
for (var i = 0; i < xs.length; i++) {
  xs[i]++
}
xs
```

#### Migrating to GraalVM JavaScript from Rhino

JavaScript scripts that have been written against Rhino may need to be updated to run properly on GraalVM JavaScript.

#### E4X

E4X is not supported by GraalJS. Scripts that use E4X should be updated to explicitly use Rhino, or be changed to not use E4X to build or parse XML.

#### Java interoperability

Both engines can interface with Java objects and classes seamlessly. However, while Rhino will allow lossy conversion, for example, converting a double value (e.g. `14.2`) to an integer value (`14`), for GraalJS a `TypeError` will be thrown.

Scripts may thus need to be updated to perform integer rounding using `Math.round`. For example, scripts in transcode presets that modify the resolution may need to be updated from:

```
var r = new com.vidispine.generated.ResolutionType();
r.setWidth(512 / 1.5);
```

To explicitly round using `Math.round`:

```
var r = new com.vidispine.generated.ResolutionType();
r.setWidth(Math.round(512 / 1.5));
```

#### Java return values

Rhino will not automatically convert value objects to the native JavaScript counterpart. GraalJS will on the other hand. For example, this will work on Rhino:

```
// the return type of getWidth is a java.lang.Long
var width = preset.getThumbnailResolution().getWidth().intValue();
```

But with GraalJS the `getWidth` method will return a JavaScript number, not a `java.lang.Long`.

### 14.5.2 Common JavaScript functions

A number of global variables are defined for the script to use. It is also possible to add custom global JavaScript objects and functions, as described in *Add generic JavaScript code*.

#### The `api` object

The `api` object can be used to perform a synchronous HTTP request to the Vidispine API. By default the request will be performed as the user that created the job that is running, unless overridden by the script using the *`api.user()`* function.

These functions all return a new `api` object with the parameters of the function added to it, and should thus be chained as shown in the example below.

`api.`**`path`**(*path*)
Adds the given path to the API URI.

**Arguments**

- **`path`** (*`string`*) – The path to add.

`api.`**`queryParam`**(*key*, *value*)
Adds a query parameter to the API URI.

**Arguments**

- **key** (*string*) – The name of the query parameter to set.
- **value** (*object*) – The value to set. Primitive types will be converted to a string.

api.**header**(*key*, *value*)
Adds a header parameter to the API URI.

**Arguments**

- **key** (*string*) – The name of the header to add.
- **value** (*string*) – The header value to add.

api.**dataType**(*type*)
The type of data that should be returned from the server.

**Arguments**

- **type** (*string*) – Supported types are `text`, `json` and `xml`, or a media type such as *application/json*. The default is `json,xml`.

api.**input**(*input*[, *type* ])
The data to be sent. The content type is optional if the input is a JavaScript or XML object, but mandatory if input is a string (such as a JSON or a XML string).

**Arguments**

- **input** (*string*) – The data to be sent.
- **type** (*string*) – Supported types are `text`, `json` and `xml`, or a media type such as *application/json*.

api.**user**(*username*[, *password* ])
The user to authenticate as. If no password is specified then the request will be authenticated using token authentication.

**Arguments**

- **username** (*string*) – The username to set.
- **password** (*string*) – The password to set.

api.**timeout**(*timeout*)
Sets the timeout of the request.

**Arguments**

- **timeout** (*long*) – The timeout in milliseconds.

Once the request parameters have been specified the request can be performed using one of these four functions:

api.**get**()
Performs a GET request.

api.**put**()
Performs a PUT request.

api.**post**()
Performs a POST request.

api.**delete**()
Performs a DELETE request.

For example, to retrieve the metadata and shapes for a specific item:

```
item = api.path("item").path(itemId)
   .queryParam("content", "metadata,shape")
   .get();
```

### Rich output

By adding `rich()` on the `api` chain, more information about the HTTP response is given. Without `rich`, the operation functions (*`api.get()`* et al.) only return the value returned by the API, and throws an exception if the API returns an error.

`api.`**`rich`**`()`

With `rich`, the functions returns a JavaScript object, with the following properties:

- `output` - The response, parsed as an object.
- `response` - The response as a string.
- `status` - The HTTP status code.
- `httpheader-*` - The various HTTP headers, with the HTTP header name in lower case, e.g. `httpheader-content-length`.

### API call information

To aid in troubleshooting API calls, this function can be used to get information about the call that is about to be made.

`api.`**`getInfo`**`()`

**Returns**

A JavaScript object with properties:

- `uri` - The URI of the request.
- `queryParams` - A `javax.ws.rs.core.MultivaluedMap` containing all of the query parameters.
- `headerCount` - Number of headers set.
- `inputIsXML` - True if the input is an XML object.
- `inputIsJSON` - True if the input is a JSON object.
- `returnTypes` - The media types that have been set using *`api.dataType()`*.
- `user` - The name of the user performing the request.
- `passwordIsSet` - True if the password has been set.

### The `http` object

The `http` object is similar to the `api` object, but can be used to invoke other HTTP resources. The `http` object needs to be used with the *`http.uri()`* function, which takes one parameter, the URI to be used.

`http.`**`uri`**`(`*uri*`)`

**Arguments**

- **`uri`** (*`string`*) – The URI of the resource.

`http.`**`followRedirects`**`(`*followRedirects*`)`

**Arguments**

- **followRedirects** (*boolean*) – If true, follows HTTP redirects. Default false.

`http.`**proxy**(*uri*)

**Arguments**

- **uri** (*string*) – Use supplied URI as HTTP proxy. See also *Proxying HTTP connection via a VSA*.

Example:

```
var uri = api.path('version').getInfo().uri;
http.uri(uri).user('admin','admin').dataType('JSON').get().licenseInfo.licenceType
```

### Proxying HTTP connection via a VSA

**Since** 21.3.

It is possible to use a VSA to proxy the HTTP request. This is very useful if the endpoint is not reachable from VidiCore, but from the VSA.

In order to proxy the connection, use `proxy()`. Multiple proxy calls can be chained, then VidiCore will select _one_ of the VSAs that is online at the moment to use as proxy.

`http.`**proxy**(*uri*)

**Arguments**

- **uri** (*string*) – The `vxa` URI of the resource.

On the VSA, the following setting is needed:

```
forwardProxy={regular expression}
```

Regular expression has to match the URI of the endpoint. In case of a `HTTPS` request, the path is not available, so the regular expression has to match an empty path.

### Example

In VSA's agent.conf:

```
forwardProxy=https?://.*
```

JavaScript code:

```
http.uri("http://localservice:7777/integration")
      .proxy("vxa://742c17e4-8f6f-489a-8caf-aae4c39d272f/")
      .proxy("vxa://e2c4c766-4a3e-4662-975c-7f4251385d8c/")
      .get()
```

### The `shell` object

The `shell` object is used to invoke shell commands.

`shell.`**exec**(*command*[, *arg*, ...][, *options*])
Executes the command with the given arguments.

**Arguments**

- **command** (*string*) – The name of the command to execute.
- **arg** (*string*) – Any arguments to pass to the command.

- **options** – A JavaScript object with fields:
  - `timeout` - Timeout in milliseconds.
  - `input` - Input to send to standard input.
  - `output` - `java.io.OutputStream` to contain the output from the command. If this field is specified then `output` will not be included in the response.
  - `err` - `java.io.OutputStream` to contain the error output from the command. If this field is specified then `output` will not be included in the response.

**Returns**

An object with fields:

- `exitcode` - The return code (an integer) from the command.
- `output` - Standard output as a string.
- `err` - Standard error as a string.

A step that checks a file for viruses might for example look something like:

```
var file = ...
var result = shell.exec("clamscan", file);
if (result.exitcode == 1) {
  job.failFatal("Virus(es) found");
}
```

### The `logger` object

The `logger` object outputs information to the log file of the application server. If the JavaScript object is concatenated to a string, the full representation may not be shown.

```
logger.log('information is '+info);
```

This can be fixed by using the *`logger.json()`* function:

```
logger.log('information is '+logger.json(info));
```

`logger.`**`log`**(*message*)
Logs the given message to the application server log file.

**Arguments**

- **message** (*object*) – The message to log. If this is a JavaScript object then it will automatically be transformed into JSON format.

`logger.`**`json`**(*object*)
Converts the given JavaScript object into JSON.

### The `metadatahelper` object

The `metadatahelper` object contains some convenient functions for generating a new metadata object.

`metadatahelper.`**`createMetadata`**()
Returns a new *MetadataType*.

`metadatahelper.`**`createMetadataTimespan`**(*start*, *end*)
Returns a new *MetadataType*.Timespan.

**Arguments**

- **start** (*string*) – The start timecode.
- **end** (*string*) – The end timecode.

`metadatahelper.`**createMetadataGroup**(*name*)
Returns a new *MetadataGroupValueType*.

**Arguments**

- **name** (*string*) – The name of the group.

`metadatahelper.`**generateMetadataField**(*name*, *value*)
Returns a new *MetadataFieldValueType*.

**Arguments**

- **name** (*string*) – The name of the field.
- **value** (*string*) – The field value.

`metadatahelper.`**metadataToStr**(*metadata*)
Translate a metadata object to a string.

**Arguments**

- **metadata** – *MetadataType*

`metadatahelper.`**log**(*obj*)
Write the value of `obj.toString()` to the server log.

The following example script

```
var metadata = metadatahelper.createMetadata();
var timespan = metadatahelper.createMetadataTimespan("0", "100");
var group = metadatahelper.createMetadataGroup("mrk_marker");
var field1 = metadatahelper.createMetadataField("mrk_color", "red");
group.getField().add(field1);
timespan.getGroup().add(group);
metadata.getTimespan().add(timespan);
```

will generate a metadata object like this:

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="0" end="100">
    <group>
      <name>mrk_marker</name>
      <field>
        <name>mrk_color</name>
        <value>red</value>
      </field>
    </group>
  </timespan>
</MetadataDocument>
```

### The `notification` object

New in version 4.17.

The `notification` can be used to trigger notifications. It is available to JavaScript job steps and from the *JavaScript test resource*.

`notification.`**send**(*notificationId*, *data*)
Trigger the notification with the given id with the specified data. Returns the notifications id.

The `data` parameter must be a JavaScript object with string keys and values that are either string or a list of strings.

**Arguments**

- **notificationId** (*string*) – The id of the notification to be used.
- **data** (*object*) – A JavaScript object with the key-value data to send.

#### Examples

For example, to trigger a notification with id `VX-45`:

```
notification.send("VX-45", {
   "hello": "world"
});
```

The notification id can also be an external id, and multiple values can also be provided:

```
notification.send("external-system", {
   "hello": "world",
   "items": ["VX-1", "VX-2"]
});
```

The data shape and delivery method and destination is decided by the notifications action. For example, if the notification `VX-45` was a HTTP notification with a content-type set to `application/xml`, that notification endpoint would receive:

```
<SimpleMetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <key>hello</key>
    <value>world</value>
  </field>
</SimpleMetadataDocument>
```

### 14.5.3 Debugging JavaScript

The JavaScript code can be debugged. When using Rhino, debugging can be done using Eclipse. GraalVM JavaScript supports debugging via the Chrome DevTools Protocol (https://chromedevtools.github.io/devtools-protocol/), using debuggers such as Chrome Developer Tools (https://developers.google.com/web/tools/chrome-devtools/).

The configuration property `debugJavaScript` controls if debugging is enabled or not. With this setting set to `true`, all JavaScript code will wait for a remote debugger to attach before continuing.

Debugging can also be enabled on a per-script basis by setting the `debug` flag in the *script header*. For example:

```
/* interpreter=graalvm, debug=true */
```

#### Debugging GraalVM JavaScript

1. **Enable JavaScript debugging**. Set the configuration property `debugJavaScript` to `true`.
2. **Execute a script**. Use *`POST /javascript/test`* to execute some JavaScript code:

   ```
   POST API/javascript/test
   Content-Type: application/javascript

   /* interpreter=graalvm */
   var a=3;
   ```

```
var b=4;
a+b;
```

If `debugJavaScript` is `true`, then the call will not return immediately.

3. **Connect the debugger**. From *GET /javascript/session*, find the Chrome Devtools debugging URL:

```
GET API/javascript/session
```

```
HTTP/1.1 200 OK
Content-Type: text/plain

0        unknown STARTED/RUNNING chrome-devtools://devtools/bundled/js_app.html?
↪ws=localhost:59000/56e2efcf-1551-48f6-ab43-f591033b5b72  null
```

Start Chrome and paste the `chrome-devtools://` in the URL bar and press Enter.

### Debugging Rhino

1. **Enable JavaScript debugging**. Set the configuration property `debugJavaScript` to `true`.

3. **Set up Eclipse**. To set up Eclipse for debugging, select *Run → Debug Configurations...* and create a new Remote JavaScript configuration. Use Mozilla Rhino as Connection, and port 59000. The port can be changed using the configuration property `debugJavaScriptPort`.

   For Source Lookup Path, select a File System Directory, and point it to any existing directory. The directory does not have to contain the source files; it will be sent via the Mozilla Rhino connector.

3. **Execute a script**. Now, Eclipse is ready to connect. Use *POST /javascript/test* to execute some JavaScript code:

```
POST API/javascript/test
Content-Type: application/javascript

var a=3;
var b=4;
a+b;
```

If `debugJavaScript` is `true`, then the call will not return immediately.

4. **Connect the debugger**. In Eclipse, choose *Run → Debug Configurations...*, select the created configuration and choose *Debug*.

   Eclipse should show a file named `testscript-xxxx.js` or similar in the source window. The first line includes the text `debugger;`. This is intentional and can be ignored; it is only added so that the debugger will actually start in suspended mode.

   Also, when the script starts, a random line – typically the second or third – is selected. Single-step once and the first line should be selected and the actual debugging can start. After the script has completed, the API call returns.

Changed in version 5.0: Support for enabling debugging via the script header was added.

## 14.5.4 Interfacing with the JavaScript engine manually

In order to test functionality, the JavaScript engine can be called manually. For more information, see *JavaScript*.

### 14.5.5 Add generic JavaScript code

In order to avoid redundant code, it is possible to register JavaScript code in a "global library". This is done using configuration properties of the form `javascript-{extension}`, where `extension` is any suffix.

When doing this, all code that is in the `javascript-` properties will be executed before the specific code. Multiple properties can be added, and will be parsed in lexical order. It is advised that only definitions (`function`) are made, and not direct statements, in order to avoid confusion.

#### Example

```
$ curl -uadmin:admin -Hcontent-type:text/plain \
localhost:8080/API/configuration/properties/javascript-1234 -X PUT --data-binary \
'function add(a,b) {
  return a+b;
}'
$ curl -uadmin:admin -Hcontent-type:application/javascript \
localhost:8080/API/javascript/test -X POST --data-binary \
'var a=3;
var b=4;
add(a,b);'
```

## 14.6 Archive Integration

Vidispine has no built in integration with any archive vendors. It is however possible to write your own integration scripts which Vidispine will then invoke when a file is to be archived.

### 14.6.1 Creating an archive storage

In order to get a working integration with an external archive, a special storage must be created with type ARCHIVE. For this integration to work, a script must be associated with the storage (described below).

#### Example

Creating an ARCHIVE storage:

```
POST /storage
```

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>ARCHIVE</type>
  <capacity>1000000000</capacity>
  <archiveScript><![CDATA[
...
]]></archiveScript>
</StorageDocument>
```

#### Integrating with an archive using JavaScript

To enable integration, a JavaScript must be written which will perform the actual archive operation. In order to be as flexible as possible, this script can both make API calls to Vidispine (*The api object*), and invoke shell operations (*The shell object*).

The script also has access to a `file` object. This object has the following functions defined:

`file.`**`getMetadata`**`(`*key*`)`
: If the specified metadata key is set on the file, the value is returned, otherwise null.

`file.`**`setMetadata`**`(`*key*, *value*`)`
Sets the specified key-value pair as metadata on the file.

`file.`**`getAllMetadata`**`()`
Returns a map of all file metadata.

The script must as its last assignment define an object with the following properties:

- `archive` - A function invoked when an archive is to be performed.
- `restore` - A function invoked when a restore is to be performed.
- `remove` - A function invoked when a delete is to be performed.
- `restorePartial` - A function invoked when a partial restore is to be performed. This function is optional. If it is missing and a partial restore is requested, the restore function will be invoked.

### Example

A simple archive script. This script only performs file system copies and removes.

```
function getFilePath(url) {
  if (url.indexOf('file:///') === 0) {
    url = url.substring(7);
  }
  if (url.indexOf('file:/') === 0) {
      url = url.substring(5);
  }
  if (url.indexOf('/C:/') === 0) {
    url = url.substring(1);
  }
  return url;
}

o = {
  "archive": function(uri, id, data) {
    var archivePath = getFilePath(data.archiveDir);
    var path = getFilePath(uri);
    var filename = uri.substring(uri.lastIndexOf('/')+1);

    logger.log('Running command "cp '+path+' '+archivePath);

    var result = shell.exec('cp', path, archivePath);
    if (result.exitcode > 0) {
        throw "Failed to copy file to archive: "+result.err;
    } else {
        file.setMetadata('uri', archivePath+filename);
    }
  },
  "restore": function(uri, id, data) {
    var path = getFilePath(uri);
    var loc = getFilePath(file.getMetadata('uri'));

    logger.log('Running command "cp '+loc+' '+path);

    var result = shell.exec('cp', loc, path);
    if (result.exitcode > 0) {
        throw "Failed to copy file from archive: "+result.err;
    }
  },
```

```
  "remove": function(id, data) {
    var loc = getFilePath(file.getMetadata('uri'));

    logger.log('Running command "rm '+loc);

    var result = shell.exec('rm', loc);
    if (result.exitcode > 0) {
        throw "Failed to remove file: "+result.err;
    }
  }
};
```

### 14.6.2 Amazon Glacier

Vidispine can archive files on Amazon Glacier. There are two different ways this can be achieved:

- Creating a separate Glacier storage and move files from other storages there for archival.
- Using an S3 storage and transition objects to the Glacier storage class.

#### Creating a dedicated Glacier storage

To create a storage used solely for Glacier archiving, you need to create a storage with an XML document like this:

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>ARCHIVE</type>
  <bean>GlacierBean</bean>
  <capacity>100000000000000</capacity>
  <metadata>
    <field>
      <key>glacierAccessKeyId</key>
      <value>{access key}</value>
    </field>
    <field>
      <key>glacierSecretKey</key>
      <value>{secret key}</value>
    </field>
    <field>
      <key>glacierVaultName</key>
      <value>{vault name}</value>
    </field>
    <field>
      <key>glacierEndpoint</key>
      <value>https://glacier.us-east-1.amazonaws.com/</value>
    </field>
  </metadata>
</StorageDocument>
```

New in version 5.3.

Credentials for the storage can now be added as metadata as show in the example above. The *secret key* will be encrypted when the storage is created or updated.

If either *glacierSecretKey* or *glacierAccessKeyId* is missing, then the credentials will be read from the `AwsCredentials.properties` file in the *credentials directory*, if one exists.

Files can then be moved to this storage either using storage rules or by initiating a copy job (*Move/copy a file to another storage*). Restore jobs must be initiated using storage rules.

Note that restore jobs typically take several hours, and the restore job will be put in the WAITING state while the restore initiation is in progress. This is to allow other jobs to run during this time. New in version 5.1.6.

Vidispine adds metadata to files stored in Amazon Glacier or Glacier Deep Archive. Using the key `s3ArchiveStorageClass`, it is possible to distinguish what storage class the file has. This metadata is then removed if the file is restored.

#### Transitioning files From S3 to Glacier

There is no way, using the AWS SDK, to directly initiate a transition to the Glacier storage class for a single object. Instead, Object Lifecycle Management (http://docs.aws.amazon.com/AmazonS3/latest/dev/object-lifecycle-mgmt.html) must be used. Vidispine will automatically detect when a transition to the Glacier class has happened, and put the file in the ARCHIVED state.

To restore a file so that it can be read directly, you can use the following request:

**PUT {file-resource}/restore**
Triggers a request to Glacier to initiate a restore.

Once the restore is complete, the file will be put in CLOSED state, and will be available for direct access.

The `expirationInDays` parameter has to be set and specifies how long the restored files should be available. Once it has expired, it will be removed from direct access and once again end up in the ARCHIVED state.

**Query Parameters**

- **extraData** (*string[]*) – Additional parameters relevant for the restore, in the form of `key=value`. Specify multiple parameters using multiple query parameters.

  **expirationInDays={number-of-days}** How long the restored files should be available.

  Deprecated since version 4.9: Use the `expirationInDays` query parameter instead.

- **expirationInDays** (*integer*) – Required. How long the restored files should be available.
- **retrievalTier** (*string*) – Sets the Glacier retrieval tier (http://docs.aws.amazon.com/AmazonS3/latest/dev/restoring-objects.html) to use when restoring the file. One of `Expedited`, `Standard` or `Bulk`.

**Produces**

- **text/plain** – Informational text.

**Status Codes**

- **400 Invalid Input** – Amazon rejects the restore request, a parameter value is invalid.
- **404 Not Found** – The file was not found.

### 14.6.3 Atempo Digital Archive Integration

Vidispine can archive from and retrieve files to any storage location which has a corresponding agent set up in an Atempo Digital Archive environment. In such a setup, each archive location would be defined as a separate storage, and any agent would also have a corresponding storage.

To set up an Atempo archive storage, use the following storage XML:

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>ARCHIVE</type>
  <bean>AtempoDigitalArchiveBean</bean>
  <capacity>100000000000000</capacity>
```

```
  <metadata>
    <field>
      <key>atempoWebServiceEndpoint</key>
      <value>http://atempo-lto/meta/C9ABD9AAD3883A8CFD3EEB922C90B3F3/721b786531/ADA/
↪WS/</value>
    </field>
    <field>
      <key>atempoRootPath/key>
      <value>/</value>
    </field>
    <field>
      <key>atempoArchiveName/key>
      <value>AMM</value>
    </field>
  </metadata>
</StorageDocument>
```

And for each storage that has a corresponding agent, the storage XML should look like this (note the storage metadata);

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>LOCAL</type>
  <autoDetect>true</autoDetect>
  <method>
    <uri>file:///mnt/storage/</uri>
    <read>true</read>
    <write>true</write>
    <browse>true</browse>
  </method>
  <metadata>
    <field>
      <key>atempoRootPath</key>
      <value>/mnt/atempo-agent1/</value>
    </field>
    <field>
      <key>atempoAgentName</key>
      <value>atempo-agent1</value>
    </field>
  </metadata>
</StorageDocument>
```

Now, to archive a file residing on the agent storage, all that is needed is to start a normal file copying job. The same mechanism is used to restore from the archive to the agent.

### 14.6.4 Front Porch Diva Integration

Set up a shared folder which is accessible by both Vidispine and DIVarchive manager.

POST the following document to /API/storage

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
 <type>ARCHIVE</type>
 <capacity>1000000000000</capacity>
 <bean>DIVABean</bean>
 <metadata>
  <field>
   <!-- SSH host -->
   <!-- this is the hostname of the DIVA SSH service -->
   <key>hostname</key>
```

```
  <value>187.47.11.109</value>
 </field>
 <field>
  <!-- SSH username -->
  <!-- this is the username for the DIVA SSH service -->
  <key>username</key>
  <value>diva</value>
 </field>
 <field>
  <!-- SSH password -->
  <key>password</key>
  <value>diva</value>
 </field>
 <field>
  <!-- SSH port -->
  <key>port</key>
  <value>22</value>
 </field>

 <field>
  <!-- path to the shared folder on vidispine server -->
  <key>storagePath</key>
  <value>/shared/storage/</value>
 </field>
 <field>
  <!-- hostname or IP address for the DIVA manager -->
  <key>DIVAHostname</key>
  <value>187.47.11.109</value>
 </field>
 <field>
  <!-- TCP port for the DIVA manager -->
  <key>DIVAPort</key>
  <value>9065</value>
 </field>
 <field>
  <!-- Media name designates either a group of tape, or an array of disk
       declared in the configuration where the instance has to be created. -->
  <key>DIVAMediaName</key>
  <value>default</value>
 </field>
 <field>
  <!-- category -->
  <key>DIVACategory</key>
  <value>default</value>
 </field>
 <field>
  <!-- restore is not yet implemented -->
  <key>DIVARestoreDestination</key>
  <value></value>
 </field>
 <field>
  <!-- path to shared folder on DIVA server -->
  <key>DIVAFilePathRoot</key>
  <value>C:/shared/storage/</value>
 </field>
 <field>
  <!-- The value of this option is the name of the source/destination to be used
       by the specified command: archive, restore, copy... This server name
```

```
        must be a valid name as configured in the DIVA system. -->
   <key>DIVAServerName</key>
   <value>disk</value>
  </field>
 </metadata>

 <method>
  <uri>file:///shared/storage/</uri>
  <read>true</read>
  <write>true</write>
  <browse>true</browse>
 </method>
</StorageDocument>
```

To archive a file, copy it to the shared folder and wait for Vidispine to detect its presence. Once Vidispine has found the file, import it to trigger archiving.

Example using curl:

```
curl -X POST -uadmin:admin 'http://localhost:8080/API/storage/VX-4/file/VX-1/import' -
↪Hcontent-type:application/xml -d '<MetadataDocument/>'
```

Archiving should begin shortly.

## 14.7 S3 Event SQS Notifications

Scanning an S3 storage can be expensive both in terms of time and money. To make it cheaper to access an S3 bucket, you can configure Vidispine to poll an Amazon SQS queue for S3 events and increase the time between regular storage scans, which are more expensive.

### 14.7.1 Prerequisites

Assuming that you already have an S3 storage setup in Vidispine, the next step is to create an SQS queue and configure the S3 bucket to send events to that queue. This configuration is made entirely within AWS and instruction on how to configure it can be found here: https://docs.aws.amazon.com/AmazonS3/latest/dev/NotificationHowTo.html

---

**Note:**

- The three types of events that Vidispine are interested in are:
  - `ObjectCreated:*` (All object create)
  - `ObjectRemoved:*` (All object delete)

  New in version 21.3.
  - `ObjectsRestore:Completed` (Object restored from archive)
- Vidispine will connect to the SQS queue using the credentials from the S3 storage method URI, so that user must have access to **both** the bucket and the queue. For the SQS queue the user needs permission for the following actions on the queue:
  - `sqs:GetQueueUrl`
  - `sqs:ReceiveMessage`
  - `sqs:DeleteMessage`
  - `sqs:DeleteMessageBatch`

  - `sqs:PurgeQueue`
- Use one SQS queue per bucket. Don't send events from multiple buckets to the same queue, as this is not supported by Vidispine.

### 14.7.2 Use IAM roles

New in version 21.3.

It is now possible to use IAM roles to handle SQS queue notifications across accounts.

- This feature will use *`stsRegion`* and use that region when making the call to the STS API for the assume role request.
- RoleSessionName is an optional field, if unset Vidicore will generate one automatically.
- Please note that the role that Vidicode will try to assume must have the same SQS permissions as above.

### 14.7.3 Close restored files faster

New in version 21.3.

Faster update of file state (CLOSED) when a file is restored from archive.

- To make Vidicore `close` files faster when they are restored from archive, make sure that `ObjectsRestore:Completed` is checked in Event types of the storage's notifications.

### 14.7.4 Configure the storage

1. To have Vidispine poll a SQS queue instead of scanning a S3 bucket, set the storage method metadata *`sqsName`* and *`sqsEndpoint`* to enable this feature:

```
PUT /storage/VX-1/method/VX-2/metadata/sqsName
Content-Type: text/plain

s3-event-queue
```

```
PUT /storage/VX-1/method/VX-2/metadata/sqsEndpoint
Content-Type: text/plain

sqs.eu-west-1.amazonaws.com
```

   New in version 21.3.

```
PUT /storage/VX-1/method/VX-2/metadata/roleArn
Content-Type: text/plain

arn:aws:iam::<accountId>:role/<roleName>
```

```
(optional)
PUT /storage/VX-1/method/VX-2/metadata/roleExternalId
Content-Type: text/plain

external-id
```

```
(optional)
PUT /storage/VX-1/method/VX-2/metadata/roleSessionName
Content-Type: text/plain
```

```
role-session-name
```

```
GET /storage/VX-1
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  ...
  <method>
    <uri>s3://bucketname/</uri>
    ...
    <metadata>
      <field>
        <key>sqsName</key>
        <value>s3-event-queue</value>
      </field>
      <field>
        <key>sqsEndpoint</key>
        <value>sqs.eu-west-1.amazonaws.com</value>
      </field>
      <field>
        <key>roleArn</key>
        <value>arn:aws:iam::<accountId>:role/<roleName></value>
      </field>
      <field>
        <key>roleExternalId</key>
        <value>external-id</value>
      </field>
      <field>
        <key>roleSessionName</key>
        <value>role-session-name</value>
      </field>
    </metadata>
  </method>
  ...
</StorageDocument>
```

2. Then make sure that the storage metadata *scanOnStart* is `true` (this is the default).

   Due to the distributed nature of Amazon SQS, the messages come unordered (http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/DistributedQueues.html). On every start up, Vidispine will need to purge the queue and do a full scan of the storage, to sync the file list with database.

3. Finally you can configure Vidispine to do regular scans of the storage less often by setting the storage property `scanInterval`. Vidispine will perform a storage scan every `scanInterval` second, so setting this to 3600 will make Vidispine scan it once every hour. See *When are files scanned?* for more information.

You can check the storage method status (`lastSuccess`, `lastFailure`, `failureMessage`) to determine if the configuration is correct or not. For example, if a non-existing queue is specified:

```
<failureMessage>
   Error polling SQS: The specified queue does not exist for this wsdl version. (...)
</failureMessage>
```

## 14.8 S3 Event SNS Notifications

New in version 5.0.

Scanning an S3 storage can be expensive both in terms of time and money. To make it cheaper to access an S3 bucket, you can configure Vidispine to receive S3 events from an Amazon SNS topic and increase the time between regular storage scans, which are more expensive.

### 14.8.1 Prerequisites

Assuming that you already have an S3 storage setup in Vidispine, the next step is to create an SNS topic and configure the S3 bucket to send events to that topic. This configuration is made entirely within AWS and instructions on how to configure it can be found here: https://docs.aws.amazon.com/AmazonS3/latest/dev/NotificationHowTo.html

---

**Note:**

- The two types of events that Vidispine are interested in are:
  - `ObjectCreated:*` (All object create)
  - `ObjectRemoved:*` (All object delete)

  New in version 21.3.
  - `ObjectsRestore:Completed` (Object restored from archive)
- The Vidispine endpoint that handles SNS messages requires authentication, so the url that is registered as a subscriber in AWS must have the credentials encoded in it. Because of this the Vidispine API must be accessible via HTTPS. Unencrypted HTTP is not supported.
- Use one SNS topic per bucket. Don't send events from multiple buckets to the same topic, as this is not supported by Vidispine.

---

### 14.8.2 Close restored files faster

New in version 21.3.

Faster update of file state (CLOSED) when a file is restored from archive.

- To make Vidicore `close` files faster when they are restored from archive, make sure that `ObjectsRestore:Completed` is checked in Event types of the storage's notifications.

### 14.8.3 Configuration

1. Set the storage method metadata *`snsTopic`* to the ARN of the SNS topic. Vidispine will only accept SNS messages from this topic.

```
PUT /storage/VX-1/method/VX-2/metadata/snsTopic
Content-Type: text/plain

arn:aws:sns:eu-west-1:123456791011:topic_name
```

```
GET /storage/VX-1
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  ...
  <method>
```

```
    <uri>s3://bucketname/</uri>
    ...
    <metadata>
      <field>
        <key>snsTopic</key>
        <value>arn:aws:sns:eu-west-1:123456791011:topic_name</value>
      </field>
    </metadata>
  </method>
  ...
</StorageDocument>
```

2. Configure Vidispine as a subscriber for this topic in AWS. The endpoint that receives SNS message is `/API/sns-endpoint`, and as this endpoint requires authentication the credentials must be encoded in the url. A complete url for subscription might look like this: `https://<username>:<password>@<address-to-vidispine>/API/sns-endpoint` or `https://sns-user:my-password@example.myvidispine.com/API/sns-endpoint`.

---

**Note:**

- The role *_file_write* is required to call the SNS endpoint. You can use a pre-existing user or create a new user specifically for the SNS subscription, as long as it has this role.
- Vidispine will automatically confirm the SNS subscription request that is sent by AWS when the subscription is created, as long as the topic matches the storage method metadata *`snsTopic`*. Checking that the subscription has been confirmed is a good way of verifying that everything has been configured correctly.

---

3. Finally you can configure Vidispine to do regular scans of the storage less often by setting the storage property `scanInterval`. Vidispine will perform a storage scan every `scanInterval` second, so setting this to 3600 will make Vidispine scan it once every hour. See *When are files scanned?* for more information.

## 14.9 Signiant Integration

Vidispine can initiate transfers between storages using Signiant. Some configuration is needed in order for this to work. Once all required configuration is set, Signiant will automatically be used for transfers between configured storages.

### 14.9.1 General system configuration

The following configuration properties must be set:

*`signiantManagerHost`* The hostname of the Signiant manager.

*`signiantManagerUser`* The manager username.

*`signiantManagerPassword`* The manager password.

### 14.9.2 Storage configuration

The following metadata field must be set on the source and destination storage (*Key-value metadata*).

`signiantAgent`

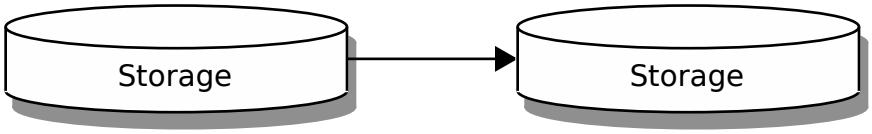
The name of the agent connected to this storage. This can also be set to an agent group using the format `<group_name>!<organization_number>`.

#### Storage methods

Each Signiant enabled storage needs a `file:/` storage method. This will be used to determine the source and destination paths for transfers.

## 14.10 Aspera Integration

Vidispine can initiate transfers between storages using Aspera. For this to work, the source and destination storages must first be configured properly.




### 14.10.1 Source storage configuration

The following metadata fields must be set on the storage (*Key-value metadata*).

**`asperaRootPath`** The absolute path to the storage root

**Example** `/usr/local/aspera-storage/`

**`asperaWsdlLocation`** The URL to the FIMS WSDL file for the Transfer Service

**Example** `http://10.18.12.10:8080/FIMS/TransferService?wsdl>`

**`asperaStatusWsdlLocation`** The URL to the FIMS WSDL for the Transfer Status Service.

**Example** `http://10.18.12.10:8080/FIMS/TransferStatusService?wsdl>`

If performing a transfer from this storage, and the destination is an Aspera URL, then Aspera will be used for the transfer.

### 14.10.2 Destination storage configuration

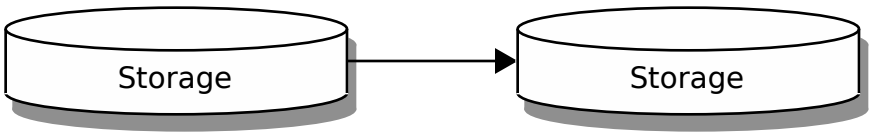
**`asperaDestinationUri`** The Aspera URI corresponding to the root folder of this storage

**Example** `fasp://10.18.12.11:22/usr/local/aspera-destination?user=username&passwo`

If a transfer is initiated to this storage, and the source storage is also configured for Aspera transfer, then Aspera will be used for the transfer.

## 14.11 Aspera FASP Integration

Vidispine can initiate transfers between storages using Aspera FASP Manager. For this to work, firstly, Aspera FASP client must be installed with a server license in order to be able to transfer any files. Secondly, the source or destination storage must be configured properly.




### 14.11.1 Transfer type

Two different transfers are supported:

1. Transfer from a local storage (with a `file` URI method) to a FASP storage.
2. Transfer from a FASP storage to a local storage (with a `file` URI method).

### 14.11.2 Storage configuration

To specify that FASP can be used to transfer from/to a storage, a special Storage Method should be added:

**URI** `fasp://{user}:{password}/{host}[:{port}]/{any relative path from the Fasp root}?{query parameters}`

**type** `TRANSFER`

The following query parameters are supported:

**`targetrate`** The target/maximum rate of the transfer, in kbit/s.

The default value is: `targetrate = 10000`

**`minrate`** The minimum rate of the transfer, in kbit/s.

The default value is: `minrate = 0`

**`udp`** FASP uses UDP and TCP ports to transfer a file.

The default value is: `udp = 33001`

**`tcp`** FASP uses UDP and TCP ports to transfer a file.

The default value is: `tcp = 22`

**`overwrite`** `overwrite` handles if the transferred file have the exact same name as another file in the destination folder.

`overwrite` can be specified to `always` (always overwrite), `never` (never overwrite), `diff` (overwrite if the content is different), `older` (overwrite if the transferred file is newer) or `diffandolder`.

The default value is: `overwrite = diff`

#### Example

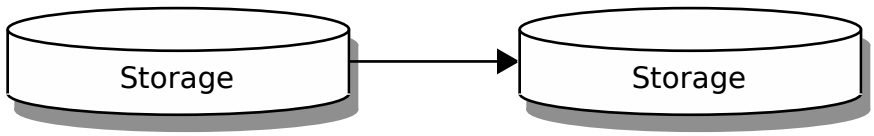
Assume that Vidispine and Aspera FASP client runs on the same server, and that you want to create a storage which can handle FASP transfers. Further assume that in FASP there is a user `usr` with password `passw` and the storage directory is at `/temp/`. Further, assume that FASP is running on port 2100.

Then the storage should look like:

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>LOCAL</type>
  <method>
    <uri>file:///temp/</uri>
    <read>true</read>
    <write>true</write>
    <browse>true</browse>
    <type>NONE</type>
  </method>
  <method>
    <uri>fasp://usr:passw@localhost:2100/temp/?targetrate=5000&amp;minrate=1000&amp;
↪udp=33002&amp;tcp=23&amp;overwrite=always</uri>
    <type>TRANSFER</type>
  </method>
</StorageDocument>
```

## 14.12 FileCatalyst Integration

Vidispine can initiate transfers between storages using FileCatalyst. For this to work, the source and/or destination storages must first be configured properly.




### 14.12.1 Transfer type

Three different transfers are supported:

1. Transfer between two storages listed in FileCatalyst Server.
2. Transfer from a local storage (with a `file` URI method) and a storage listed in FileCatalyst Server.
3. Transfer from a storage listed in FileCatalyst Server to a local storage.

### 14.12.2 Storage configuration

To specify that FileCatalyst can be used to transfer from/to a storage, a special Storage Method has to be added:

**URI** `filecatalyst://{user}:{password}/{host}[:{port}]/{any relative path from the FileCatalyst root}`

**type** `TRANSFER`

#### Example

Assume that Vidispine and FileCatalyst runs on the same server, and that you want to create a storage to handle `/srv/media/incoming`. Further assume that in FileCatalyst there is a user `fc` with password `s3cret` which has is FileCatalyst home root directory at `/srv/media`. Further, assume that FileCatalyst is running on port 2100.

Then the storage should look like:

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>LOCAL</type>
  <method>
    <uri>file:///srv/media/incoming/</uri>
    <read>true</read>
    <write>true</write>
    <browse>true</browse>
    <type>NONE</type>
  </method>
  <method>
    <uri>filecatalyst://fc:s3cret@localhost:2100/incoming/</uri>  <!-- /srv/media +␣
↪incoming = /srv/media/incoming -->
    <type>TRANSFER</type>
  </method>
</StorageDocument>
```

## 14.13 MXFserver Integration

The MXFserver plugin allows Vidispine to integrate with MXFserver. The plugin allows collections to be created in Vidispine, representing business units, sections, modules, episodes and projects (here called **entities**.) Items added to a project collection will automatically be added to the MXFserver project.

### 14.13.1 Set up

The plugin can also extend *LDAP* so that business units and sections are created for imported users.

A JDBC resource should be configured in your application server for connecting to the MXFserver MySQL database. The `databaseName` element can be used to specify the JNDI name of the JDBC resource (default= `jdbc/mxfserver`). Typical connection pool settings are:

| General settings | |
|---|---|
| Datasource Classname | com.mysql.jdbc.jdbc2.optional.MysqlDataSource |
| Resource Type | javax.sql.DataSource |

| Additional properties | |
|---|---|
| Url | jdbc:mysql://<mxfserver-ip-addess>:3307/system5 |
| User | <username> |
| Password | <password> |

#### Requirements

- The MySQL JDBC driver (http://www.mysql.com/products/connector/) , installed into GlassFish (`$GLASSFISH/lib`).

#### Installation

1. Configure the plugin by creating a MXFserver resource containing the MXFserver settings, by making a `POST` request to `API/resource/mxfserver` containing:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <mxfserver>
    <url>http://192.168.38.200:11000/mxfserver/</url>
    <workspaceUrl>file:///mnt/mxfserver/</workspaceUrl>
    <mxfServerWorkspacePath>C:\storage\Workspaces\</mxfServerWorkspacePath>
```

```
    <mxfServerUserId>1</mxfServerUserId>
    <mxfServerPathToStorage>C:\storage\Vidispine\</mxfServerPathToStorage>
    <storageId>VX-1</storageId>
    <db-host>192.168.38.200</db-host>
    <db-port>3307</db-port>
    <db-username>root</db-username>
    <db-password>Mastermeta</db-password>
    <atomShapes>atom</atomShapes>
    <importShapes>hd,original</importShapes>
  </mxfserver>
</ResourceDocument>
```

In the example above the (first) shape with the `hd` tag should be imported, if one exists, and the `original` shape should be used otherwise.

The elements are:

**mxfServerWorkspacePath** The path to the workspaces directory used by MXFserver.

**mxfServerUserId** The Vidispine MXFserver user id.

**mxfServerPathToStorage** The path to the Vidispine storage `storageId` as seen by MXFserver.

**storageId** The storage that contains the files that should be imported into MXFserver. Must be on the same file system as the MXFserver workspaces directory.

**atomShapes** Should contain the tags of the shapes that contain OP-Atoms, and for which QuickTime reference files will be created.

**importShapes** Contains the shapes that should be considered for import into a MXFserver project, ordered by priority.

2. Enable the plugin by setting the following configuration properties:

| Property | Value | Note |
|---|---|---|
| `collectionPluginBean` | `MxfServerCommunicator` | |
| `ldap.import.plugins` | `MxfServerUserImportPlugin` | |
| `ldap.attr.businessUnit` | `Department` | |
| `ldap.attr.section` | `UoSCourse` | |
| `ldap.groupsAsFunctions` | `TRUE/FALSE` | (1) |

(a) If enabled then MXFserver functions will be created with the same names as the groups found in the directory.

3. To enable automatic import of new media added to projects, the MXFserver configuration file `Mxfserver.ini` needs to be updated with:

```
[API_OPTIONS]
notifyNewFilesHTTPLocation=http://[Vidispine server address]/MxfServerAPI/import
```

### 14.13.2 Usage

MXFserver entities are in Vidispine simply created as collections. A number of additional parameters are required, depending on the type of entity to create, as shown below. See *Collections* for more on how to manage collections.

Note that a collection will automatically be added as a child to the parent collection.

The query parameters are:

**name={collection-name}** The name of the collection and MXFserver entity.

**type={entity-type}** The type of MXFserver entity. Either `businessUnit`, `section`, `programme`, `episode` or `project`.

**parent={parent-collection-id}** The id of the parent collection/entity.

**projectType={project-type}** The type of MXFserver project.

**projectBaseId={project-base-id}** The project template to extend.

### Example

Creating the MXFserver project hierarchy.

```
POST /API/collection/?name=NameOfBusinessUnit&type=businessUnit
```

```
POST /API/collection/?name=NameOfSection&type=section&parent=VX-1
```

```
POST /API/collection/?name=NameOfProgram&type=programme&parent=VX-2
```

```
POST /API/collection/?name=NameOfEpisode&type=episode&parent=VX-3
```

With a FCP (`projectType=2`) project based on the FCP7 template ( `projectBaseId=30`).

```
POST /API/collection/?name=NameOfProject&type=project&parent=VX-4&projectType=2&
↪projectBaseId=30
```

**Caution:** Note that projects can be created at any level in the hierarchy. However, the MXFserver client only allows projects to be created if an episode has been selected. An episode must also be selected before the details of a project can be edited and saved, which if done would cause Vidispine to be out of sync with MXFserver (it's one way sync from VS to MXFserver only.)

## 14.14 EVS IP Director Integration

It is possible to map data inside "log info" (<Log>) in a EVS metadata file to Vidispine metadata.

### 14.14.1 Example

Import the EVS metadata file as a sidecar file with your essence file:

```
POST /import?URL=/vidispine/demo.dv&sidecar=file:///path/to/evs.metadata.xml"
```

so a EVS metadata that looks like this:

```
<EVS_Metadatas Revision="1">
  <General_Infos>
                ...
  </General_Infos>
  <Clips_Infos>
    <Clip>
      <XFile_Clip_Infos>
                                ...
      </XFile_Clip_Infos>
      <Other_Clip_Infos>
                                ...
```

```
        <Logs>
          <Log DBVersion="0" GUID="2b9de077-8e4d-4e48-ac8f-b2cdc05b0805" Version="2.0.
↪1">
            <Date>21-Apr-2013</Date>
            <TC>15:00:33:01 </TC>
            <DateUser>21-Apr-2013</DateUser>
            <TCUser>15:00:33:01 </TCUser>
            <TCTable>1</TCTable>
            <Description>Mål av: 11. Selakovic, Stefan</Description>
            <TapeID />
            <InterestLevel>0</InterestLevel>
            <Colour>0</Colour>
            <AvidColour>#000000</AvidColour>
            <Keywords>
              <Keyword Type="Keyword">Mål</Keyword>
              <Keyword Type="Keyword">HBK</Keyword>
              <Keyword Type="Participant">11. Selakovic, Stefan</Keyword>
            </Keywords>
            <AutomaticKeywords>
              <AutomaticKeyword Description="" Header="Attendance" Type="NUMBER">4011
↪</AutomaticKeyword>
              <AutomaticKeyword Description="" Header="Away Team" Type="TEXT">Kalmar␣
↪FF</AutomaticKeyword>
              <AutomaticKeyword Description="" Header="HalfTimeScore" Type="TEXT">1-0
↪</AutomaticKeyword>
           </AutomaticKeywords>
          </Log>

      </Other_Clip_Infos>
    </Clip>
  </Clips_Infos>
</EVS_Metadatas>
```

will be translated to Vidispine metadata like:

```
<?xml version="1.0"?>
<timespan start="1350826@PAL" end="1350851@PAL">
  <group uuid="a0c2d689-bee3-48ea-8708-5228e533382c" user="admin" timestamp="2013-11-
↪29T11:50:21.938+01:00" change="VX-8131">
    <name>EVS_Log</name>
    <field uuid="2eb192e7-1af8-4cde-9083-93e5c4c922bd" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
      <name>EVS_AvidColour</name>
      <value uuid="2d060045-bd11-4d17-bba2-5327d51d3ee7" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">#000000</value>
    </field>
    <field uuid="9efabf9f-003d-437a-a459-3c1f9a4a306e" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
      <name>EVS_Colour</name>
      <value uuid="014a932c-bfc3-4a36-a209-c4f9f3389b0b" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">0</value>
    </field>
    <field uuid="48323112-c06b-4ef3-855f-550f422f5d83" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
      <name>EVS_InterestLevel</name>
      <value uuid="264c55a3-4679-439d-ac2a-dd63f7e57b93" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">0</value>
    </field>
```

```
    <field uuid="f035327f-c006-49cb-8ed3-2d4c78fd35e7" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
      <name>EVS_TapeID</name>
      <value uuid="81cc0f37-cf8d-4b3c-9641-a94367aa4a1d" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131"/>
    </field>
    <field uuid="ba4bc026-5900-4215-be94-515b4568379a" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
      <name>EVS_Description</name>
      <value uuid="ed800484-c646-46b7-8530-e6487f2dc637" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">Mål av: 11. Selakovic, Stefan</value>
    </field>
    <field uuid="3742dfeb-a100-4980-9db5-d3281342b9a8" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
      <name>EVS_TCTable</name>
      <value uuid="06490341-dae1-42d7-a81a-abea7015bcb3" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">1</value>
    </field>
    <field uuid="09f4af36-05f2-43f5-a063-ddc680ef18f4" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
      <name>EVS_TCUser</name>
      <value uuid="5e38b2cd-3d06-4a5a-8358-b6da51a58637" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">15:00:33:01 </value>
    </field>
    <field uuid="48df81fc-4ba3-4ad8-847f-22521a0ae89b" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
      <name>EVS_Date</name>
      <value uuid="bfc7b8e1-aa98-4f32-ad50-30e899c67834" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">21-Apr-2013</value>
    </field>
    <field uuid="b397b900-d869-4457-9351-e08fb53a5670" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
      <name>EVS_TC</name>
      <value uuid="45a6e2f3-2111-45ee-aa38-4373d710bc29" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">15:00:33:01 </value>
    </field>
    <field uuid="87993aa8-6c61-49c8-a834-fb73649db7b7" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
      <name>EVS_DateUser</name>
      <value uuid="ea2b8cdd-4e91-4468-bbe7-00342f194ecd" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">21-Apr-2013</value>
    </field>
    <group uuid="d6db1d19-4bb5-41cc-a01b-faaa41eadf13" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
      <name>EVS_Keywords</name>
      <field uuid="26755143-ad34-45b5-a68f-e0d6407beb5a" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
        <name>EVS_Keyword</name>
        <value uuid="eb83876d-0852-4b82-912d-4468324ff5e7" user="admin" timestamp=
↪"2013-11-29T11:50:21.938+01:00" change="VX-8131">11. Selakovic, Stefan</value>
      </field>
      <field uuid="1b8f7603-3623-4ae7-9b18-ac363973c2ae" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
        <name>EVS_Keyword</name>
        <value uuid="812085e9-175e-44f0-a74e-b27dec67dd33" user="admin" timestamp=
↪"2013-11-29T11:50:21.938+01:00" change="VX-8131">HBK</value>
      </field>
      <field uuid="acb9a9b6-06d1-4e9a-8838-ab7efac97b59" user="admin" timestamp="2013-
↪11-29T11:50:21.938+01:00" change="VX-8131">
```

```
        <name>EVS_Keyword</name>
        <value uuid="bd148e7d-f7d3-446d-8147-6af9cee23127" user="admin" timestamp=
↪"2013-11-29T11:50:21.938+01:00" change="VX-8131">Mål</value>
      </field>
    </group>
  </group>
</timespan>

<timespan start="-INF" end="+INF">
  <group uuid="90cf23d4-2555-48b5-a38c-f284076f8cdd" user="admin" timestamp="2013-11-
↪29T11:50:23.783+01:00" change="VX-8131">
    <name>EVS_MatchData</name>
    <field uuid="868c16d8-22cf-41d8-a2fe-bccc4221d686" user="admin" timestamp="2013-
↪11-29T11:50:23.783+01:00" change="VX-8131">
      <name>EVS_HalfTimeScore</name>
      <value uuid="ad12c72c-2030-41e9-81e0-7605869f501d" user="admin" timestamp="2013-
↪11-29T11:50:23.783+01:00" change="VX-8131">1-0</value>
    </field>

    <field uuid="374c7c03-b54c-46fe-81ea-b52eef353fc2" user="admin" timestamp="2013-
↪11-29T11:50:23.783+01:00" change="VX-8131">
      <name>EVS_AwayTeam</name>
      <value uuid="4082bbfd-e875-445f-9baa-beec03cb6e5e" user="admin" timestamp="2013-
↪11-29T11:50:23.783+01:00" change="VX-8131">Kalmar FF</value>
    </field>
    <field uuid="a97e745a-fcbb-42a3-b0f2-fc73b27ae7b9" user="admin" timestamp="2013-
↪11-29T11:50:23.783+01:00" change="VX-8131">
      <name>EVS_Attendance</name>
      <value uuid="e17dd5e3-f85c-4477-afc1-170c1d7f3d71" user="admin" timestamp="2013-
↪11-29T11:50:23.783+01:00" change="VX-8131">4011</value>
    </field>

  </group>
</timespan>
```

Please note that the values in `<AutomaticKeyword>`s will be mapped as global metadata (with timespan: (-INF, +INF)), so it is a good place to store the metadata of the whole essence file.

## 14.15 StorNext Integration

In version 4.2.3, beta support for Quantum StorNext file information is added. With this support, storage information from StorNext is added to the file information. StorNext version 5 and higher is supported, and the Web Services API and the HTTP protocol must be enabled.

### 14.15.1 Storage configuration

In order for StorNext information to be retrieved, a special Storage Method has to be added:

**URI** `stornext://{user}:{password}/{host}:{port}/{StorNext path}`

**type** `HSM`

Here, user and password are the StorNext web services API credentials. (`webservice`, `webservice` by default on StorNext). Host and port is the StorNext endpoint (typically port 81). StorNext path is the base path prefixed to the file path when the StorNext API is queried. Typically, this is the same path as for the `file` method.

#### Example

Below is an example of a storage configuration where StorNext and Vidispine runs on the same machine, with the StorNext filesystem on `/stornext/snfs`.

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>LOCAL</type>
  <method>
    <uri>file:///stornext/snfs/</uri>
    <read>true</read>
    <write>true</write>
    <browse>true</browse>
    <type>NONE</type>
  </method>
  <method>
    <uri>stornext://webservice:webservice@localhost:81/stornext/snfs/</uri>
    <type>HSM</type>
  </method>
</StorageDocument>
```

### 14.15.2 StorNext Metadata

When the StorNext endpoint is set up, Vidispine file archive status and metadata is updated.

- The file is marked as ARCHIVED if and only if StorNext `location` is exactly `TAPE`.
- The StorNext metadata fields `location`, `class`, `existingCopies`, and `targetCopies` are set on the file. This can be changed by modifying the configuration property *`stornextFileMetadata`*. It should be a comma separated list of StorNext metadata fields.

## 14.16 Cerify integration

The Cerify plugin allows Vidispine to integrate with Cerify from Tektronix. The plugin allows video files to be analyzed by Cerify during their import. RAW_IMPORT PLACEHOLDER_IMPORT ESSENCE_VERSION and AUTO_IMPORT are supported.

### 14.16.1 Installation

1. Configure Cerify. The minimum configuration required is the creation of a `MediaLocation` with a path that is shared between Cerify and Vidispine storage, and the creation of a Profile. The profile may be empty.
2. Configure the plugin by creating a Cerify resource containing the Cerify settings, by making a `POST` request to `API/resource/cerify` containing:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <cerify>
    <address>http://cerifyserver.example.com:80/CeriTalk?wsdl</address>
    <mediaLocation>
      <name>Name of Media Location</name>
      <storageMethod>VX-6</storageMethod>
    </mediaLocation>
    <cleanup>false</cleanup>
  </cerify>
</ResourceDocument>
```

   The elements are:

**address** The URL of the Cerify web service.

**mediaLocation** One or many media locations. If many media locations are configured, the storage method where the file is stored will determine which one to use.

**name** The name of the media location. A media location with this name must be configured in Cerify.

**storageMethod** The storage method that contains the files that should be analyzed by Cerify. Must be on a file system accessible by Cerify through the path configured in the corresponding media location.

**cleanup** If set to true, jobs and media sets will be removed from Cerify after completion.

3. Update the task definition document by inserting a Cerify step at the appropriate position by making a `POST` request to `API/task-definition` containing something similar to:

```
<TaskDefinitionListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <task>
    <description>Executing Cerify job</description>
    <extradata>f</extradata>
    <flags>12</flags>
    <bean>CerifyJobBean</bean>
    <method>analyzeFile</method>
    <step>250</step>
    <dependency>
      <step>0</step>
      <previous>false</previous>
      <allPrevious>true</allPrevious>
    </dependency>
    <parallelDependency>
      <step>0</step>
      <previous>false</previous>
      <allPrevious>false</allPrevious>
    </parallelDependency>
    <jobType>RAW_IMPORT</jobType>
    <cleanup>false</cleanup>
    <critical>true</critical>
  </task>
</TaskDefinitionListDocument>
```

Note: for ESSENCE_VERSION, the Cerify job step should run after step 400; for AUTO_IMPORT, the Cerify step should run after step 200.

### 14.16.2 Usage

The Cerify profile to use when analyzing a file is specified using the `jobmetadata` query parameter.

Import a file and let Cerify analyze it using the Cerify profile named `mpeg2 PAL` :

```
curl -X POST -u admin:admin --data-binary @test_file.mpg 'http://127.0.0.1:8080/API/
↪import/raw?throttle=false&jobmetadata=cerifyProfile%3Dmpeg2%20PAL'
```

For AUTO_IMPORT job, the job metadata can be set in the AutoImportRuleDocument. For example:

```
<AutoImportRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <tag>mp4</tag>
  <jobmetadata>
    <field>
      <key>cerifyProfile</key>
      <value>Vidispine test profile</value>
    </field>
```

```
  </jobmetadata>
</AutoImportRuleDocument>
```

When the file is being analyzed by Cerify there will be progress information available in the job. The metadata key is `cerifyProgress` and the value will be an integer between 0 and 100.

Use the `cerifyPriority` job metadata field to set the Cerify job priority (LOW, MEDIUM, HIGH). For example:

```
curl -X POST -u admin:admin --data-binary @test_file.mpg 'http://127.0.0.1:8080/API/
↪import/raw?throttle=false&jobmetadata=cerifyProfile%3Dmpeg2%20PAL&
↪jobmetadata=cerifyPriority%3DHIGH'
```

### 14.16.3 Output

Upon completion the results from Cerify is added to the shape as bulky metadata. The following fields are available. Note that `cerify_alerts` might not always be present and its absence means that Cerify did not detect any problem with the file.

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>cerify_alert</uri>
  <uri>cerify_jobinfo</uri>
  <uri>cerify_streaminfo</uri>
</URIListDocument>
```

The element `cerify_alerts` contains all alerts produced by Cerify. Example:

```
<field start="466@3082500:128557" end="466@3082500:128557">
  <key>cerify_alert</key>
  <maps>
    <map>
      <entry key="alertFrame">http://10.185.0.7:80/ViewFrame.do?&amp;jobmediafile=115&
↪amp;frame=467&amp;audio=false&amp;small=true</entry>
      <entry key="alertId">22015</entry>
      <entry key="details">In Main profile / Main level, the maximum permitted value␣
↪of f_code[0][1] in a frame picture is 5. In the current picture it has been set to␣
↪7. Stream position: 0x7cf812 (dec. 8189970), bit 7 Bitstream context: [VSQ|PCX]</
↪entry>
      <entry key="level">error</entry>
      <entry key="location">00:00:15;16 frame 467</entry>
      <entry key="title">Invalid f_code</entry>
      <entry key="trackId">-1</entry>
      <entry key="type">video</entry>
      <entry key="url">http://10.185.0.7:80/protected/AlertDetails.do?job=107&amp;
↪jobmediafile=115&amp;frame=467&amp;alertid=1383&amp;trackId=-1</entry>
    </map>
  </maps>
</field>
```

`cerify_streaminfo` contains general information about the analyzed file, such as peak volume level, frame rate, etc.

See the following documents for for more complete examples of metadata documents produced by this plugin:

- `cerify_alert.xml`
- `cerify_jobinfo.xml`
- `cerify_streaminfo.xml`

## 14.17 FIMS implementation

Vidispine implements the FIMS 1.0.7 Transform specification (http://wiki.amwa.tv/ebu/index.php/SPECIFICATIONS). Apart from the mandatory features, Vidispine also supports the following optional features:

- *Notifications* - Vidispine will send HTTP callbacks for events such as job success and job failure.
- *Job priorities* - Jobs can be assigned one of five priorities.

The services are available at the following location: http://localhost:8080/FIMS/TransformMediaService

### 14.17.1 Codecs and formats

Vidispine currently supports a subset of the container formats and codecs defined by EBU. For a full list of formats, see:

- Container formats specified by EBU (http://www.ebu.ch/metadata/cs/web/ebu_ContainerFormatCS_p.xml.htm).
- Video codecs specified by EBU (http://www.ebu.ch/metadata/cs/web/ebu_VideoCompressionCodeCS_p.xml.htm).
- Audio codecs specified by EBU (http://www.ebu.ch/metadata/cs/web/ebu_AudioCompressionCodeCS_p.xml.htm).

#### Container formats

The following container formats are supported in Vidispine.

| ID | Name |
|---|---|
| 7.1.2, 7.2.2.2 | MP4 |
| 7.2.3 | DV |
| 7.2.4 | AVI |
| 7.2.11 | MOV |
| 7.2.19 | MKV |
| 7.2.15 | FLV |
| 7.2.7, 7.2.8 | ASF/WMV |
| 7.3.1, 7.3.1.3 | JPG |
| 7.3.8 | PNG |

#### Video codecs

The following video codecs are supported in Vidispine.

| ID | Name |
|---|---|
| 2 | MPEG-2 |
| 5 | MJPEG |
| 6 | JPG |
| 9 | H264 |
| 10 | VC-1 |
| 20 | DVVIDEO |
| 28 | VP8 |

#### Audio codecs

The following audio codecs are supported in Vidispine.

| ID | Name |
|---|---|
| 7.3, 8.4 | MP3 |
| 7.2 | MP2 |
| 8 | AAC |
| 11 | PCM_S16LE |
| 19 | WMAv1 |

## 14.18 CloudConvert Integration

Vidispine supports using CloudConvert (https://cloudconvert.com/) as an alternative to the Vidispine transcoder.

**Important:**

- It is not possible to transcode using the Vidispine transcoder and CloudConvert at the same time.
- Thumbnails and posters will not be generated during the transcode of video using CloudConvert

### 14.18.1 How to use

1. Create a CloudConvert resource on Vidispine, containing your CloudConvert API key:

```
POST /resource/cloudconvert
```

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <cloudconvert>
    <apiKey>your-api-key</apiKey>
  </cloudconvert>
</ResourceDocument>
```

2. Start the jobs with job metadata `useCloudConvert=true`.

```
POST /item/(item-id)/transcode?jobmetadata=useCloudConvert=true
```

### 14.18.2 Source file access

By default, Vidispine will upload the source file to CloudConvert. However, it's also possible to let CloudConvert fetch them directly.

Firstly, make sure the source files are publicly available:

- If the files are on S3 or Azure, set the configuration property `useS3Proxy` or `useAzureProxy` to `true`;
- If not, a public address of the Vidispine server is needed in the CloudConvert resource:

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <cloudconvert>
    <apiKey>your-api-key</apiKey>
    <publicAddress>http://public-address:8080</publicAddress>
  </cloudconvert>
</ResourceDocument>
```

  Otherwise, the address from the configuration property `apiNoauthUri` will be used.

Then, set the job metadata `cloudConvertInputMethod=download` or add the global setting in the CloudConvert resource:

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <cloudconvert>
    <apiKey>your-api-key</apiKey>
    <publicAddress>http://public-address:8080</publicAddress>
    <inputMethod>download</inputMethod>
  </cloudconvert>
</ResourceDocument>
```

### 14.18.3 Conversion parameters

When transcoding, CloudConvert specific options can be specified in the `<muxerSetting>` of the shape tag. For example:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>jpg</format>
  <video>
    <codec>jpg</codec>
  </video>
  <audio/>
  <muxerSetting>
    <key>converteroptions[resize]</key>
    <value>300x300</value>
  </muxerSetting>
  <muxerSetting>
    <key>converteroptions[quality]</key>
    <value>80</value>
  </muxerSetting>
</TranscodePresetDocument>
```

### 14.18.4 CloudConvert callback

After a job has been started, Vidispine will always try to poll CloudConvert for the job status. In order for Vidispine to receive notifications (https://cloudconvert.com/api/conversions#callback) from CloudConvert, the `publicAddress` of the Vidispine server should be set:

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <cloudconvert>
    <apiKey>your-api-key</apiKey>
    <publicAddress>http://public-address:8080</publicAddress>
  </cloudconvert>
</ResourceDocument>
```

If not set, then the `apiNoauthUri` will be used. The one resource that must be accessible to CloudConvert is the *callback* resource.

### 14.18.5 Enable CloudConvert using JavaScript

JavaScript can be used to determine if CloudConvert should be used in a job. The script will be evaluated if the job metadata `useCloudConvert` is not set. CloudConvert will be used only if the script returns a string `true`.

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <cloudconvert>
    <apiKey>your-api-key</apiKey>
    <script><![CDATA[
        ...
 ]]></script>
```

```
  </cloudconvert>
</ResourceDocument>
```

The available objects are: *job* and the *commonly available objects*

## 14.19 EIDR Integration

Vidispine support importing and synchronizing EIDR-metadata into the Vidispine metadata model. It keeps your metadata up to date with the EIDR records and allows transforming EIDR metadata into Vidispine metadata.

Read more about EIDR here. (http://eidr.org/)

### 14.19.1 Setup

Create an EIDR resource:

```
POST /resource/eidr HTTP/1.1
Content-Type: application/xml

<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <eidr>
    <url>https://resolve.eidr.org/EIDR</url>
    <include>eidr_base</include>
    <include>eidr_credits</include>
  </eidr>
</ResourceDocument>
```

Make sure the URL is pointing to the EIDR web API endpoint. `resolve.eidr.org` is the public endpoint, you can also set it to the sandbox endpoints for testing purposes.

If you have a registered account, you can provide credentials, by including all of the three elements `userId`, `partyId`, `password`. The `passwordShadow` is built by hashing the password with md5 and encoding the resulting bits with base64:

```
passwordHash = base64( md5(raw-password) )
```

```
POST /resource/eidr HTTP/1.1
Content-Type: application/xml

<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <eidr>
    <url>https://resolve.eidr.org/EIDR</url>
    <include>eidr_base</include>
    <include>eidr_credits</include>
    <userId>userID</userId>
    <partyId>partyID</partyId>
    <password>passwordShadow</password>
  </eidr>
</ResourceDocument>
```

`include` are projections that transform EIDR metadata to Vidispine compatible metadata documents. Provided includes are:

| include | Description |
|---|---|
| `eidr_base` | Titles, release date, origin etc. |
| `eidr_credits` | Directors and actors |

`include` defaults to `eidr_base` if none is given.

It's possible to customize what data is included by creating your own incoming XSLT projections, the projection should result in a valid standalone metadata document that could be used for placing metadata on an item. See *Metadata projections* and *Metadata projections* for how to add new projections.

### 14.19.2 EIDR synchronization

In the following reference, `{eidr-content-resource}` is one of the following:

- `/item/(item-id)`
- `/library/(library-id)`

#### Synchronize EIDR metadata

**PUT `{eidr-content-resource}/eidr/sync`**
Synchronizes `item(s)` metadata that are out of date. An item is considered out of date if the EIDR record has changed or if the included projections have changed.

---

**Important:** For an item to be able to synchronize it needs the metadata field `eidr_id` with the value of an EIDR id.

---

**Caution:** It's ill-advised to synchronize a large library manually as the operation isn't asynchronous.

Returns a list of synchronized items with the metadata that was written to the item.

**Query Parameters**

- **`eidrResource`** (*`string`*) – If set, the resource identified by this id will be used instead of first found EIDR resource.
- **`forceSync`** (*`boolean`*) –
  - `true` - force metadata write to item.
  - `false` - (default)

**Produces**

- **`application/xml, application/json`** –
  - MetadataListDocument

**Role**

- _metadata_write
- _library_read, for library

#### Example

```
PUT /item/(item-id)/metadata HTTP/1.1
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan start="-INF" end="+INF">
         <field>
            <name>eidr_id</name>
```

```
            <value>10.5240/116F-AC63-B1B0-C538-CF6C-D</value>
         </field>
   </timespan>
</MetadataDocument>
```

### 14.19.3 Troubleshooting

You can test your custom include projection by using *Create a metadata change set* and put a FullMetadata xml from EIDR in the body. Example of a FullMetadata xml (https://resolve.eidr.org/EIDR/object/10.5240/5D7B-C695-C27F-E175-8455-K/?type=Full&followAlias=true)

CHAPTER

FIFTEEN

# TROUBLESHOOTING AND OBTAINING INFORMATION

## 15.1 Self test

The Vidispine self test will generate a brief report about the system infrastructure for simple troubleshooting.

### 15.1.1 Tests

The tests are:

- `api` - The API test. Verifies both `api` and `apinoauth` resources.
- `solr` - Verifies the Solr configuration.
- `database` - Verifies that the database can be reached.
- `transcoder` - Verifies that the transcoder can be reached.
- `jms` - Verifies that the JMS queues are well configured.
- `tools` - Verify the existence of various external tools.
- `simplejob` - Verifies that the transcoder can execute a simple transcode job.
- `thumbnail` - Verifies the thumbnail configuration.
- `dbstats` - Returns some statistics from the database schema.
- `ldap` - Verifies the LDAP configuration.
- `fileaccesstest` - Verifies write access of application server files.
- `cluster` - Verifies the status of cluster instances and lists its members.

### 15.1.2 Test results

There are four possible outcomes of a test:

**OK** Vidispine is well configured and running properly.

**Warning** Some configuration is not valid.

**Failed** There are tools missing, which could lead to failures of some functions.

**Critical** Important configuration or tools are missing or invalid, Vidispine will not run properly.

### 15.1.3 Running the test

Running the tests can take a while, depending on the size of the system. The `dbstats` is typically the slowest as it examines the number of rows in the database among other things. Hence, it sometimes can be beneficial to only run certain specific tests.

Use the *self test resource* to execute the tests. For example:

```
GET API/selftest
```

```
<SelfTestDocument xmlns="http://xml.vidispine.com/schema/vidispine" status="failed"␣
↪took="38170ms">
  <test name="api" status="warning" took="408ms">
    <test name="adminApi" status="warning">
      <message>API is 4.2</message>
      <message>Transcoder VX-1 is ERROR: Could not connect</message>
      <message>Transcoder VX-2 is 4.1.4-gea8cc32-12511</message>
    </test>
    <test name="apiNoAuth" status="ok">
      <message>Base uri: http://localhost:8089/</message>
    </test>
  </test>
  ...
</SelfTestDocument>
```

## 15.2 Error log report

The error log report is used to collect information about the Vidispine installation, and contains information about certain jobs or items. A log report should always be included if you encounter an issue that you wish to report to us .

### 15.2.1 Usage

This tool is located at [your server]:8080/LogReport. After filling in all the information, press *Extract and collect logs* and wait while the system extract the needed logs; this might take a while. Then press on 'Save report' and save the created .zip-file on your computer. Then send this file to your Vidispine reseller.

#### Required fields

**Error report information** This is where you describe your problem. Be specific on what the problem is, what you did when it appeared.

**Time span** What time did the error occurred? Set start time and end time

**Credentials** Your Vidispine user-name and password

**Optional fields** Fill in this information if any is applicable on the particular problem you are having..

**Job-ID** The id of the job that failed, if applicable to the issue.

**Item-ID** The id of the item that the issue relates to, if applicable.

**Storage-ID** The id of the storage that the issue relates to, if applicable.

**User** Name of the Vidispine user that was used when this problem occurred.

### 15.2.2 Programmatically retrieving log files

If your application has a custom form for reporting issues then you can instead collect the log files using the *Vidispine logs resource*.

For example, to retrieve a log report for a specific job:

```
GET API/vidispine-logs?job=VX-32&comment=Incorrect%20aspect%20ratio%20of%20transcoded
 ↪%20image
Accept: application/zip
```

```
200 OK
Content-Type: application/zip
Transfer-Encoding: chunked

...
```

CHAPTER

SIXTEEN

# INSTALLATION

The Vidispine API can either be installed on-premise or be run as a service on Vidinet (http://vidinet.net/).

This chapter details how to install Vidispine using the Debian or RPM packages.

## 16.1 Installing distribution-specific packages

Use our packages for your distribution to install Vidispine.

### 16.1.1 Install the packages

You can either install the packages from our repository, or download and install the packages from our download page.

**Note:** Vidispine requires Java 11 which may not be available in the official repositories of older distributions. Install a third party repository providing Java 11 or upgrade to a newer distribution.

#### Install from official repository

To install Vidispine directly from our repository, head over to the repository (http://repo.vidispine.com/) page and follow the instructions.

### 16.1.2 Initialize the database

1. Create and give Vidispine access to an empty database:

```
$ psql -c "CREATE USER vidispine PASSWORD 'vidispine'";
$ psql -c "CREATE DATABASE vidispine OWNER vidispine";
```

   On MySQL, make sure to use UTF-8:

```
CREATE DATABASE vidispine CHARSET utf8 COLLATE utf8_bin;
```

   On Microsoft SQL Server create the database with a case-sensitive collation. As Vidispine consequently is using unicode (NVARCHAR) columns you do not need to care about the character set or code page.

```
CREATE DATABASE vidispine COLLATE Latin1_General_CS_AS
```

   Snapshot isolation must be enabled for Vidispine to operate properly. Execute these commands directly after creating the database:

```
ALTER DATABASE vidispine SET ALLOW_SNAPSHOT_ISOLATION ON
GO
ALTER DATABASE vidispine SET READ_COMMITTED_SNAPSHOT ON
GO
```

Ensure that TCP/IP is enabled for your server.

2. Modify the configuration file accordingly:

```
$ vi /etc/vidispine/server.yaml
```

PostgreSQL:

```
database:
  driverClass: org.postgresql.Driver
  url: jdbc:postgresql://localhost/vidispine
  user: vidispine
  password: vidispine
```

MySQL:

```
database:
  driverClass: com.mysql.jdbc.Driver
  url: jdbc:mysql://localhost/vidispine
  user: vidispine
  password: vidispine
```

Microsoft SQL Server:

```
database:
  driverClass: com.microsoft.sqlserver.jdbc.SQLServerDriver
  url: jdbc:sqlserver://localhost:1433;instanceName=SQLEXPRESS;
↪databaseName=vidispine
  user: vidispine
  password: vidispine
```

3. Initialize and migrate the database:

```
$ vidispine db ping # verify connection
$ vidispine db check # verify if tables exists (they shouldn't)
$ vidispine db migrate
$ vidispine db check # should succeed
```

**Note:** The `/usr/bin/vidispine` command is simply an alias to `java -jar /usr/share/vidispine/server/vidispine-server.jar` that is provided by the `vidispine-server` package.

### 16.1.3 Start the services

1. Use systemd to start the services:

```
$ systemctl start solr transcoder vidispine
```

2. Wait for Vidispine to start and then run APIinit to create the system metadata fields and the admin user.

```
$ # wait for 8080 to become available, and then
$ curl -X POST localhost:8080/APIinit
```

**Note:** APIinit is a migration step that must be run manually. It will be made part of the migration command in the future.

3. To verify that Vidispine is running, access http://localhost:8080/API/version using curl or HTTPie (https://github.com/jakubroztocil/httpie), or directly in your browser. The default admin password is `admin`.

```
$ curl -X GET "localhost:8080/API/version" -uadmin:admin
```

#### Troubleshooting

- If the Vidispine service fails, then check the syslog or journal for errors:

```
$ less /var/log/syslog
```

```
$ journalctl -xn
```

- If the service dies, or never becomes available for some reason, then check the server log:

```
$ less /var/log/vidispine/server.log
```

- If the service fails to start with a "UnsupportedClassVersionError: Unsupported major.minor version 51.0", then make sure that the default system Java version is 7+ and not 6 or lower.

```
$ sudo update-alternatives --config java
```

### 16.1.4 Configure Vidispine

Finally, you will need to *configure Vidispine*.

## 16.2 Quick setup

Before using Vidispine, make sure to create and configure a storage and thumbnail location, and to configure the transcoder.

1. Create a *storage*.

```
POST API/storage
Content-Type: application/xml

<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>LOCAL</type>
  <method>
    <uri>file:///path/to/files/</uri>
    <read>true</read>
    <write>true</write>
    <browse>true</browse>
    <type>NONE</type>
 </method>
 <autoDetect>true</autoDetect>
</StorageDocument>
```

2. Create a *thumbnail resource*.

```
POST API/resource/thumbnail
Content-Type: application/xml

<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <thumbnail>
    <path>file:///path/to/thumbnails/</path>
  </thumbnail>
</ResourceDocument>
```

3. Configure a *trancoder*. For example, with Vidispine and the transcoder on the same server:

```
POST API/resource/transcoder
Content-Type: application/xml

<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <transcoder>
    <url>http://localhost:8888/</url>
  </transcoder>
</ResourceDocument>
```

**Tip:** Use curl, HTTPie (https://github.com/jakubroztocil/httpie) or the HTTP client of your choosing to make the requests. For example, using HTTPie:

```
$ http post "localhost:8080/API/storage" @storage.xml
$ http post "localhost:8080/API/resource/thumbnail" @thumbnail.xml
$ http post "localhost:8080/API/resource/transcoder" @transcoder.xml
```

## 16.3 Service configuration

### 16.3.1 The `vidispine` service user

The post-installation script in the packages will create the `vidispine` user and group if they do not exist.

If you want to make sure that the `vidispine` user has a specific UID and GID, then create the `vidispine` user and group manually *before* installing any packages.

### 16.3.2 Service dependencies

If you're running all components on the same server, then it can be beneficial to make sure that the transcoder and Solr are started before Vidispine.

Create a systemd unit.d directory (https://www.freedesktop.org/software/systemd/man/systemd.unit.html) and override the service dependencies in a drop-in file:

```
$ vi /etc/systemd/system/vidispine.service.d/local.conf
[Unit]
Wants=transcoder.service solr.service

$ systemctl daemon-reload
$ systemctl enable vidispine
```

### 16.3.3 Setting JVM options

Create a systemd drop-in file and override the default JAVA_OPTS:

```
$ vi /etc/systemd/system/vidispine.service.d/local.conf
[Service]
Environment="JAVA_OPTS=-Xmx8192m -XX:MaxPermSize=512m"
```

## 16.4 Clustering

A cluster can be created by installing Vidispine on multiple servers and configuring all instances to connect to the same database.

The one setting that should be set is the `bindAddress`, which an instance will bind to and publish to the other members of the cluster.

```
cluster:
  bindAddress: vs1.example.com
```

You can also change the address that is published, for example if there's a firewall with port forwarding rules set up in front of each server.

```
cluster:
  bindAddress: vs1.example.com
  bindPort: 7800
  bindPortRange: 0
  externalAddress: fw.example.com
  externalPort: 7801
```

---

**Note:** For this to work you also need to use an external ActiveMQ instance, so make sure that the embedded broker is disabled and that the configuration points to your ActiveMQ instance:

```
broker:
  url: tcp://activemq.example.com:61616
  #embeddedBroker: broker:(tcp://localhost:61616)
```

---

### 16.4.1 Quick cluster setup

It is also possible to create a cluster on a single machine by starting multiple server processes each with a different configuration file.

```
$ cp server.yaml instanceA.yaml instanceB.yaml
$ vi instanceA.yaml instanceB.yaml
```

Make sure that all instances have distinct ports. Then start the instances that are to be part of the cluster:

```
$ java -jar vidispine-server.jar server instanceA.yaml 2>&1 1>instanceA.log &
$ java -jar vidispine-server.jar server instanceB.yaml 2>&1 1>instanceB.log &
```

Tail the log and you should see that the processes have found each other and have formed a cluster.

```
INFO  [2015-05-27 13:06:27,652] [403] org.infinispan.remoting.transport.jgroups.
↪JGroupsTransport: ISPN000094: Received new cluster view: [di2-37999|1] [di2-37999,␣
↪di2-55654]
```

## 16.5 Upgrading

### 16.5.1 Upgrading Vidispine

1. Install the latest server and transcoder packages.
2. Use the check command to verify that the configuration file is still valid.

```
$ vidispine check
$ vidispine db check
```

3. Make sure that the server is stopped.

```
$ systemctl stop vidispine
```

4. Migrate the database:

```
$ vidispine db migrate
```

   Before migrating you can verify the pending migrations using the `--dry-run` flag.

```
$ vidispine db migrate --dry-run
```

   **Note:** The dry-run output is for informational use only. Do not execute any SQL statements in the output directly.

5. Start all Vidispine services.
6. Wait for Vidispine to start and then run APIinit to create any new system metadata fields or task definitions.

```
$ # wait for 8080 to become available, and then
$ curl -X POST localhost:8080/APIinit
```

### 16.5.2 Upgrading to Vidispine 5.0

Because of a change in our internal system for the handling of database migrations, all users **MUST** upgrade to version 4.17 of Vidispine server and complete the database migration process before attempting to upgrade to Vidispine Server 5.0.

To upgrade to 5.0 from 4.17, just follow the normal *upgrade steps*.

#### Solr

The Solr version has been updated from 4.10 to 8.1. The Lucene index written by the previous version is not compatible with that of the newer version. This means that a full re-index is required to rebuild the search index.

The location of the Solr index has changed from `/var/lib/vidispine/solr/collection1/` to `/var/lib/vidispine/solr/8.x/collection1/` to avoid having Solr read the old index.

After upgrading, the old index can be removed.

#### ActiveMQ

ActiveMQ has been updated from 5.10 to 5.15. Any installations of ActiveMQ that use JDBC persistence may need to clear the ActiveMQ database tables as the format of the messages in the `activemq_msgs` table has changed and can not be consumed by the new ActiveMQ version (AMQ-5995 (https://issues.apache.org/jira/browse/AMQ-5995)).

Another option is to change ActiveMQ to persist data in different tables by setting a table prefix in the `activemq.xml`. For example:

```
<persistenceAdapter>
  <jdbcPersistenceAdapter>
    <statements>
      <statements tablePrefix="amq515_"/>
    </statements>
  </jdbcPersistenceAdapter>
</persistenceAdapter>
```

If KahaDB is used then ActiveMQ should still be able to read the existing transaction log:

```
INFO  [2019-10-02 07:25:08,843] [1] org.apache.activemq.broker.BrokerService: Using␣
↪Persistence Adapter: KahaDBPersistenceAdapter[/var/lib/vidispine/activemq]
INFO  [2019-10-02 07:25:09,169] [1] org.apache.activemq.store.kahadb.MessageDatabase:␣
↪KahaDB is version 6
WARN  [2019-10-02 07:25:09,428] [1] org.apache.activemq.store.kahadb.KahaDBStore:␣
↪Existing Store uses a different OpenWire version[6] than the version configured[11]␣
↪reverting to the version used by this store, some newer broker features may not␣
↪work as expected.
INFO  [2019-10-02 07:25:09,526] [1] org.apache.activemq.store.kahadb.plist.
↪PListStoreImpl: PListStore:[/var/lib/vidispine/server/activemq-data/localhost/tmp_
↪storage] started
```

## 16.6 Server configuration

Dropwizard 2.0 is used to start and configure Jetty and to parse and validate the command line and configuration file. This is only mentioned here as the Dropwizard Configuration Reference lists and explains the base settings that can be used in the configuration file.

- Dropwizard 1.0 Configuration Reference (http://dropwizard.github.io/dropwizard/1.0.0/docs/manual/configuration.html)

Changed in version 5.0: The Dropwizard version was updated from 1.0 to 2.0.

### 16.6.1 Environment variables

Environment variables can be used in the YAML configuration file.

For example:

```
database:
  driverClass: org.postgresql.Driver
  url: jdbc:postgresql://${DATABASE_HOST}/${DATABASE_NAME}
  user: ${DATABASE_USER}
  password: ${DATABASE_PASSWORD}
```

### 16.6.2 Additional settings

In addition, the following properties are supported:

#### database

- `type`: If this is a master/read-write or read-only database. Either `master` (default), `read_only_replica`, `read_only_snapshot`.

### api

- `requestLogging`: If *request logging* is enabled. Default is `true`.
- `searchHistory`: If *search history* is updated with new searches. Default is `true`.
- `convertMatrixParameters`: This allows matrix parameters to be translated to query parameters. Default is `true`

Changes in version 5.0:

- The `convertMatrixParameters` setting was added.
- The `openApi2Compatible` setting was removed.
- The `requestLogging` and `searchHistory` settings were added.

### secrets

The location of external private keys and credentials. Used to authenticate against storages and other endpoints. See *Storage credentials* for more information. Example:

```
secrets:
  keyStore:
    path: /env/vidispine/server.keystore
    password: changeit
  vault:
    address: http://vault.example.com:8200
    token: 2262e94c-39c3-b9a8-605d-f0450dfc558b
    keyPrefix: secret/
  file:
    path: /etc/secrets/
```

### Java KeyStore

Java Keystore (https://en.wikipedia.org/wiki/Keystore) containing private keys.

- `keyStore`:
  - `path`: The path to the Java keystore.
  - `password`: The key store password.

### Vault

A HashiCorp Vault (https://www.vaultproject.io/) server storing private keys and credentials. Private keys will be read from the `private_key` field. Usernames and passwords from the `username` and `password` fields respectively.

- vault:
  - `address`: The HTTP URL to the Vault server.
  - `token`: The Vault access token.
  - `keyPrefix`: Optional prefix to use with all aliases.

### Local file system

Read private keys and credentials from files on the local file system.

Private keys will be read from the file with the same name as the private key alias. Usernames and passwords will be read from the files `username` and `password` from the directory with the same name as the *alias*.

- file:
  - `path`: The directory containing secret files.

### broker

Configures how to connect to ActiveMQ.

- `user`: The user to authenticate as.
- `password`: The password to authenticate using.
- `url`: Default is `tcp://localhost:61616`.
- `embeddedBroker`: The broker URI (http://activemq.apache.org/broker-uri.html) to use to start an embedded broker. For example `broker:(tcp://localhost:61616)`. Default is "" (no embedded broker).

---

**Note:** If you are using embedded ActiveMQ with KahanDB, the KahanDB journal log could keep growing if there are expired messages in the queue "ActiveMQ.DLQ".

To fix this, you will need to enable JMX in the broker URI (http://activemq.apache.org/broker-uri.html) , and purge the queue manually using activemq-admin (http://activemq.apache.org/activemq-command-line-tools-reference.html).

```
embeddedBroker: broker:(tcp://localhost:61616)?usekahadb=true&kahadb.directory=/path/
↪to/db/&persistent=true&useJmx=true
```

```
./activemq-admin -Dactivemq.jmx.url=service:jmx:rmi:///jndi/rmi://localhost:1099/
↪jmxrmi purge ActiveMQ.DLQ
```

Or setup a standalone ActiveMQ instance, and set *processExpired="false"*

http://activemq.apache.org/message-redelivery-and-dlq-handling.html

---

### ejbPool

These settings configures the stateless container in OpenEJB. They are explained in more detail at http://tomee.apache.org/containers-and-resources.html.

- `maxSize`: The maximum number of beans in the stateless bean pool. Default is 10.
- `idleTimeout`:
- `strictPooling`: If the pool may NOT grow larger then `maxSize`. Default is false.

### transaction

New in version 5.4.

These settings configure the transaction manager.

- `defaultTimeout` - The default transaction timeout to use for transactions (in seconds). Default is 600.

### cluster

- `bindAddress`: The address to bind on, as an IP address or hostname. Default is 127.0.0.1.
- `bindPort`: The port to bind on. Default is 7800.
- `bindPortRange`: The range of ports to try in case `bindPort` is taken. Default is 30.
- `externalAddress`: The address to publish to members in the cluster. Default is `bindAddress`.

- `externalPort`: The port to publish to members in the cluster. Default is the port that was bound on.

### services

Background tasks, such as jobs and storage polling, are executed in the background by internal services in Vidispine.

The services that are allowed to run on this server instance can be configured using the `enabled` or `disabled` settings. These are mutually exclusive, meaning that you cannot specify both services to enable and disable at the same time.

- services:
  - `enabled`: The services that are allowed to run on this instance. All other services will not be allowed to run.
  - `disabled`: The services that are not allowed to run on this instance. All other services will be allowed to run.

The following values are supported:

- `all`: All services in the system.
- `job`: The services that execute Vidispine jobs.
- The name of a specific *Vidispine service*.

For example, an instance that should only serve API requests could be configured using:

```
services:
  disabled: all
```

### javascript

New in version 5.0.

Settings related to JavaScript script execution.

- `javascript`:
  - `bindHost`: The host that should be used when debugging JavaScript. This setting takes precedence over the `debugJavaScriptPort` configuration property. Default is `localhost` if `port` is specified, else it's null.
  - `port`: The port that should be used when debugging JavaScript. This setting takes precedence over the `debugJavaScriptPort` configuration property. Default is 59000 if `bindHost` is specified, else it's null.

### search

Configure the backend that will be used for searching.

- `backend`:
  - `solr`: (default) Use Solr as the search backend.
  - `elasticsearch`: Use Elasticsearch as the search backend.
- `url`: The URL to Elasticsearch's RESTful interface.
- `user`: The username used to connect to Elasticsearch.
- `password`: The password used to connect to Elasticsearch.

For example, to use Elasticsearch as the search backend:

```
search:
  backend: elasticsearch
  url: http://localhost:9200/
```

The `elasticsearch` backend can be configured to support two-way SSL, if client authentication (https://www.elastic.co/guide/en/elasticsearch/reference/current/security-settings.html#http-tls-ssl-settings) is enabled on the Elasticsearch cluster.

```
search:
  backend: elasticsearch
  url: https://localhost:9200/
  clientAuth:
    keyStore: /path/to/your/keystore.pfx
    keyStoreType: PKCS12
    keyStorePassword: vidispine
    alias: key-alias
```

- `keyStore`: Path to the KeyStore file.
- `keyStoreType`: The KeyStore type.
- `keyStorePassword`: The password of the KeyStore.
- `alias`: Alias of the KeyStore entry to use (Optional).

### httpClient

New in version 4.15.

These settings configure how HTTP requests are made. This includes requests made to for example:

- Storages using HTTP/HTTPS
- HTTP notification endpoints
- Transcoders
- Vidinet

### apm

New in version 5.1.

Changed in version 5.4: The `maxSpans` setting was added.

Configure application performance monitoring using Elastic APM (https://www.elastic.co/products/apm).

```
apm:
  elastic:
    urls: ["https://localhost:1234/"]
    secretToken: secret
    serviceName: vidispine
    serviceVersion: 5.0
    environment: staging
    sampleRate: 1
    maxSpans: 500
```

- `apm.elastic`:
  - `urls`: One or more URLs to the Elastic APM server(s).
  - `secretToken`: The secret token used to authenticate with the APM server(s).

  - `serviceName`: Groups all traces together by labeling using a common service name. It must conform to the following regular expression: `^[a-zA-Z0-9 _-]+$`.
  - `serviceVersion`: An optional, arbitrary, version string of the application.
  - `environment`: An optional, arbitrary, string describing the environment of the application.
  - `sampleRate`: An optional number between 0.0 and 1.0 which controls the amount of traces sent to the APM servers. 1.0 means that all traces are sent. Consider lowering this value to reduce the overhead of the tracing. Defaults to 1.0.
  - `maxSpans`: An optional number which controls the amount of spans that is sent to the APM servers. In case of transactions that create a lot of spans e.g. lot of SQL queries then this might need to be increased to be able to see all of them. Defaults to 500.

### TLS

Configure the server certificates to trust (the trust store) and the client certificates and keys to use (the key store) when using TLS client authentication.

Note that, if no `tls` field is present in the configuration file, then by default all server certificates will be trusted. However, if a `tls` fields is defined then the defaults defined in the configuration reference apply (`trustSelfSignedCertificates=false` for example).

- `tls`: The TLS settings. See the Dropwizard HTTP client TLS configuration (https://www.dropwizard.io/1.2.0/docs/manual/configuration.html#man-configuration-clients-http-tls) for a list of available properties.

For example:

```
httpClient:
  tls:
    keyStorePath: /etc/vidispine/keystore.jks
    keyStorePassword: ueLom5Oh
    trustStorePath: /etc/vidispine/truststore.jks
```

### VSA

New in version 5.1.

Changed in version 21.3.

```
vsaconnection:
  bindPort: 8183
  clusterConnect: false
  enabled: true
  forwardPortRange: 45700-45899
```

- `vsaconnection`:
  - `bindPort`: Port to bind to for SSH-connection of VSA. The same a `port` in *Register a server agent*.
  - `clusterConnect`: Set this to `true` for clustered environments. The VSA will connect to one VidiCore instance, and the other nodes will connect to the first one.
  - `enabled`: If VSAs should be allowed to connect to VidiCore.
  - `forwardPortRange`: Local port numbers for ports that are forwarded by the VSA connection. In a clustered environment, these ports must be reachable for other nodes in the cluster.

## 16.7 Package reference

### 16.7.1 Packages

The packages provided by Vidispine are:

**vidispine-server** The Vidispine server application.

**vidispine-solr** The latest supported version of Apache Solr, bundled with the Solr config and schema used by Vidispine.

**transcoder** The Vidispine transcoder.

### 16.7.2 Optional packages

**vidispine-tools** Optional command line tools.

**vidispine-server-matrixstore3.1** Must be installed to be able to connect to MatrixStore 3.1. The MatrixStore 3.1 SDK is NOT compatible with MatrixStore 3.2.

**vsctl** Optional command line tool.

### 16.7.3 Files

The key files used by Vidispine:

**/etc/vidispine/server.yaml** The server configuration file.

**/etc/vidispine/License.lic** Your Vidispine license.

**/etc/vidispine/slaveAuth.lic** Your slave license file when using master/slave licensing.

**/usr/share/vidispine/server/lib/ext/** The location of any custom JAR files.

**/var/lib/vidispine/activemq/** The default location of the ActiveMQ data files when running an embedded broker.

**/var/lib/vidispine/solr/** The location of the Solr cores.

**/var/log/vidispine/server.log** The Vidispine server log file.

**/var/log/vidispine/transcoder.log** The Vidispine transcoder log file.

CHAPTER

SEVENTEEN

# API REFERENCE

## 17.1 Access controls

Manage access to items, collections and libraries.

### 17.1.1 Managing access controls

In the following reference, `{access-entity}` is one of the following:

- `/item`
- `/collection`
- `/library`

#### Retrieve the access control list for an entity

**GET {access-entity}/**(*entity-id*)**/access**
Retrieves the entire access control list for the specified entity.

**Query Parameters**

- **additionalUserInfo** (*boolean*) – Set to `true` to return full user information instead of only username. Requires role *_user_read*. Default is `false`.

  (New in 5.3.1.)

- **additionalGroupInfo** (*boolean*) – Set to `true` to return full group information instead of only group name. Requires role *_group_read*. Default is `false`.

  (New in 5.3.1.)

**Produces**

- **application/xml, application/json** – AccessControlListDocument
- **text/plain** – CRLF-delimited list of ids

**Role** _accesscontrol_read

**See also:**

Access controls can also be viewed by recipient user, see *access control entries by user*

#### Create an access control entry

**POST {access-entity}/**(*entity-id*)**/access**
Adds a new access control entry for the specified entity.

**Query Parameters**

- **allowDuplicate** (*boolean*) – Set to `false` in order to avoid adding a duplicate access control. Default is `true`.

**Accepts**

- **application/xml, application/json** – AccessControlDocument

**Produces**

- **application/xml, application/json** – AccessControlDocument
- **text/plain** – The id of the created entry.

**Role** _accesscontrol_write

### Example

```
POST /item/VX-123/access/
Content-Type: application/xml

<AccessControlDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <permission>READ</permission>
   <group>testGroup</group>
   <operation>
      <uri/>
   </operation>
</AccessControlDocument>
```

### Retrieve an access control entry

**GET {access-entity}/**(*entity-id*)**/access/**
*access-id* Retrieves the desired access control entry.

**Query Parameters**

- **additionalUserInfo** (*boolean*) – Set to `true` to return full user information instead of only username. Requires role *_user_read*. Default is `false`.

  (New in 5.3.1.)

- **additionalGroupInfo** (*boolean*) – Set to `true` to return full group information instead of only group name. Requires role *_group_read*. Default is `false`.

  (New in 5.3.1.)

**Status Codes**

- **404 Not found** – No entry with that id exists in that entity.

**Produces**

- **application/xml, application/json** – An AccessControlDocument containing the requested access control entry.
- **text/plain** – The id of the entry.

**Role** _accesscontrol_read

### Delete an access control entry

**DELETE {access-entity}/**(*entity-id*)**/access/**
*access-id* Removes the desired access control entry.

**Status Codes**

- **200 OK** – The entry was successfully removed.
- **404 Not found** – No entry with that id exists in that entity.

**Role** _accesscontrol_write

#### Add access control entries to all items

**POST /item/access**
Adds access control entries to all known items.

**Accepts**

- **application/xml, application/json** – AccessControlDocument

**Role** _administrator

#### Delete all access control entries from all items

**DELETE /item/access**
Deletes all access control entries from all known items.

**Role** _administrator

#### Update the owner of an entity

**PUT {access-entity}/**(*entity-id*)**/access/owner/**
*username* Update the owner of the specified entity to the specified user.

**Produces**

- **application/xml, application/json** – AccessControlDocument

**Role** _administrator

### 17.1.2 Managing access controls in bulk

#### Create multiple entry access control entries

**POST {access-entity}/**(*entity-id*)**/access/bulk**
Adds multiple new access control entries to the specified entity.

**Accepts**

- **application/xml, application/json** – AccessControlListDocument

**Produces**

- **application/xml, application/json** – AccessControlListDocument
- **text/plain** – The ids of the created entries.

**Role** _accesscontrol_write

#### Example

```
POST /item/VX-123/access/bulk
Content-Type: application/xml

<AccessControlListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <access>
    <permission>READ</permission>
    <group>testGroup</group>
```

```
    <operation>
      <uri/>
    </operation>
  </access>
  <access>
    <permission>READ</permission>
    <user>testUser</user>
  </access>
</AccessControlListDocument>
```

```
<AccessControlListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <access id="VX-1043">
    <loc>http://localhost:8080/API/item/VX-123/access/VX-1043</loc>
    <grantor>admin</grantor>
    <appliesTo>all</appliesTo>
    <permission>READ</permission>
    <group>testGroup</group>
    <operation>
      <uri/>
    </operation>
  </access>
  <access id="VX-1044">
    <loc>http://localhost:8080/API/item/VX-123/access/VX-1044</loc>
    <grantor>admin</grantor>
    <appliesTo>all</appliesTo>
    <permission>READ</permission>
    <user>testUser</user>
  </access>
</AccessControlListDocument>
```

#### Delete multiple access control entries

**DELETE {access-entity}/**(*entity-id*)**/access/bulk**

Deletes multiple access control entries by id.

**Accepts**

- **application/xml, application/json** – AccessControlListDocument

**Role** _accesscontrol_write

#### Example

```
DELETE /item/VX-123/access/bulk
Content-Type: application/xml

<AccessControlListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <access id="VX-1043"/>
  <access id="VX-1044"/>
</AccessControlListDocument>
```

```
200 OK
```

### 17.1.3 Default access controls

Each user can specify what access control that will be applied to an imported item. The user importing the item will always be granted OWNER permissions.

#### List the default access controls for the current user

**GET /import/access**
Lists the access control list that will be applied on imported items.

**Produces**

- **application/xml, application/json** – An ImportAccessControlListDocument

**Role** _import

#### Example

```
GET /import/access
```

```
<ImportAccessControlListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <group>
      <name>mygroup</name>
      <permission>READ</permission>
   </group>
</ImportAccessControlListDocument>
```

#### Add a group to the default access control list

**PUT /import/access/group/**(*group-name*)
Sets the permissions for a certain group.

**Query Parameters**

- **permission** (*string*) – Required. The level of permissions to grant the group.

**Role** _import

#### Example

```
PUT /import/access/group/mygroup?permission=READ
```

```
200 OK
```

#### Remove a group from the default access control list

**DELETE /import/access/group/**(*group-name*)
Removes the specified group from the default access control list.

**Role** _import

#### Example

```
DELETE import/access/group/mygroup
```

```
200 OK
```

### 17.1.4 Viewing applied access controls

To review all access control entries that affects an item an AccessControlMergedDocument can be retrieved.

### List the applied access control entries for an entity

**GET {access-entity}/**(*entity-id*)**/merged-access/**

Retrieves a list of all access control entries that affects each user for a given entity. This includes all access derived from the user's group memberships, and from the entity's inclusion in collections or libraries.

There are two modes of operation, either retrieving the access on the item for all users or querying for the access of a specific user. In the former case no parameters are specified and in the latter all parameters must be supplied.

The `access` element of the AccessControlMergedDocument includes the following fields:

- `username` - The username of the user affected by the entry.
- `group` - The group name if the entry is derived from access granted to a group.
- `collection/library` - The collection name or library ID If the access is derived from the entity's inclusion in a collection or library.
- `rank` - The access entries for each user are ranked based on the criteria described in *Access Control Priority*. The matching entry with the lowest value for each user applies.

  Changed in version 5.3: `rank` changed name from `priority` to avoid confusion with field of the same name in AccessControlDocument.

- `matches` - This shows if the entry matches the operation specified by the `type` query parameter. If query parameters are not specified, the type defaults to `GENERIC`.
- `effectivePermission` - The effective permission if it is different from the permission initially assigned to the entry. This might occur if the grantor's permission is lower than the permission they assigned.
- `originalDisabledGrantor` - Access can be inherited through a chain of multiple grantors. If the entry is disabled due to a disabled grantor, this shows the original grantor that needs to be re-enabled for the entry to be re-enabled.

The entries are listed grouped by user, in order of rank for each user.

**Query Parameters**

- **username** (*string*) – The name of the user to check.
- **permission** (*string*) – The lowest required permission level.
- **type** (*string*) – The type of operation to check for.
- **extradata** (*string*) – Any possible extradata.

**Produces**

- **application/xml, application/json** – An AccessControlMergedDocument containing all access control that affects the entity.

**Role** _accesscontrol_read

### Example: retrieving all entries

```
GET /item/VX-1/merged-access
```

```
<AccessControlMergedDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <access rank="1" id="VX-100" username="admin">
    <permission>ALL</permission>
    <type>GENERIC</type>
  </access>
```

```
  <access rank="2" id="VX-101" username="admin">
    <permission>WRITE</permission>
    <type>GENERIC</type>
    <collection>VX-10</collection>
  </access>
  <access rank="4" id="VX-102" username="admin">
    <permission>ALL</permission>
    <type>GENERIC</type>
    <collection>VX-12</collection>
  </access>
  <access rank="1" id="VX-103" username="testUser">
    <permission>READ</permission>
    <type>METADATA</type>
    <group>mygroup</group>
  </access>
  <access rank="2" id="VX-104" username="testUser">
    <permission>READ</permission>
    <type>METADATA</type>
    <group>mygroup</group>
  </access>
</AccessControlMergedDocument>
```

### Example: querying about specific access

Checking if the user admin has full access to the metadata of item VX-1. Notice that the access provided by VX-101 is of type `SHAPE`, and so does not match the query for `METADATA` access. However, the rank of VX-101 is ranked lower than the access of VX-100 and thus the user has full access to the metadata.

```
GET /item/VX-1/merged-access?username=admin&permission=ALL&type=METADATA
```

```
<AccessControlMergedDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <query>
    <username>admin</username>
    <permission>ALL</permission>
    <type>METADATA</type>
    <item>VX-1</item>
  </query>
  <access rank="1" matches="true" id="VX-100">
    <permission>ALL</permission>
    <type>GENERIC</type>
  </access>
  <access rank="2" matches="false" id="VX-101">
    <permission>WRITE</permission>
    <type>SHAPE</type>
    <collection>VX-10</collection>
  </access>
  <access rank="3" matches="true" id="VX-102">
    <permission>ALL</permission>
    <type>GENERIC</type>
    <collection>VX-23</collection>
  </access>
  <access rank="4" matches="true" id="VX-103">
    <permission>ALL</permission>
    <type>GENERIC</type>
    <collection>VX-12</collection>
  </access>
  <access rank="5" matches="true" id="VX-104">
```

```
    <permission>ALL</permission>
    <type>GENERIC</type>
    <collection>VX-10</collection>
  </access>
</AccessControlMergedDocument>
```

### List the applied access control entries that affects groups

**GET** **{access-entity}/** (*entity-id*) **/merged-access/group**

Lists groups that have access to an entity.

Even though a user belongs to a group that has access to an entity, the user may not have access due to other access control entries that take precedence.

Groups without users will not appear, unless the group belongs to an inheritance hierarchy that has users.

**Query Parameters**

- **full** (*boolean*) –
  - `true` - Return all access controls that apply for a group. Also include additional information about the access controls in the response.
  - `false` (default) - Return a single access entry with the permission that applies for each group and type.
- **emptyGroups** (*boolean*) –
  - `true` - Include groups without users in the response.
  - `false` (default) - Do not include groups without users in the response.

**Produces**

- **application/xml, application/json** – An AccessControlMergedGroupDocument.

**Role** _accesscontrol_read

### Example

```
GET /item/VX-1000/merged-access/group
```

```
<AccessControlMergedGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <access>
    <group>groupA</group>
    <permission>READ</permission>
    <type>GENERIC</type>
  </access>
  <access>
    <group>_transcoder</group>
    <permission>WRITE</permission>
    <type>GENERIC</type>
  </access>
  <access>
    <group>_special_all</group>
    <permission>WRITE</permission>
    <type>GENERIC</type>
  </access>
  <access>
```

```
    <group>groupD</group>
    <permission>READ</permission>
    <type>GENERIC</type>
  </access>
  <access>
    <group>groupC</group>
    <permission>READ</permission>
    <type>GENERIC</type>
  </access>
  <access>
    <group>groupB</group>
    <permission>READ</permission>
    <type>GENERIC</type>
  </access>
</AccessControlMergedGroupDocument>
```

### 17.1.5 Access visualization

In order to easily see the access control that apply for an entity there is a functionality to render the access control inheritance as a graph. In order to render the graph, the Graphviz (http://www.graphviz.org/) package is required.

#### Retrieve the access graph

**GET** **{access-entity}/** (*entity-id*) **/access/graph**
Shows the entity and any ancestor collections or libraries and the access controls on each. The access-entity can be item or collection (library is not implemented).

**Query Parameters**

- **type** (*string*) –
  - ancestor - Show the entity and ancestor entities in a hierarchy.
  - grant - Show users and how permissions have been granted.
- **users** (*boolean*) – If the user and group hierarchy should be shown as a subgraph for the relevant users/groups.
- **groups** (*boolean*) – If groups should be shown as nodes in the grant graph.

**Produces**

- **image/png** –

**Role** _administrator

#### Retrieve the access graph as dot file

**GET** **{access-entity}/** (*entity-id*) **/access/graph/dot**
Shows the entity and any ancestor collections or libraries and the access controls on each. The access-entity can be item or collection (library is not implemented).

**Query Parameters**

- **type** (*string*) –
  - ancestor - Show the entity and ancestor entities in a hierarchy.
  - grant - Show users and how permissions have been granted.
- **users** (*boolean*) – If the user and group hierarchy should be shown as a subgraph for the relevant users/groups.

- **groups** (*boolean*) – If groups should be shown as nodes in the grant graph.

**Produces**

- **text/plain, text/vnd.graphviz** –

**Role** _administrator

## 17.2 Audit trails

The audit log records all requests made to the API, excluding the request data, for later use. It is typically used for troubleshooting, to be able to determine what happened when, and for examining actions taken by users or other services.

### 17.2.1 Examining the log

#### List all audit log entries

**GET /log**

Retrieves log entries according to the specified filtering criteria. The path can be seen as having an implicit wildcard in the end, unless it is disabled with the `wildcard` parameter. For example `/item/VX-123` will match `/item/VX-123/shape` but not `/item/VX-124`.

Changed in version 5.1: The body parameter was added.

Changed in version 21.3: The `path` query parameter now supports wildcards, ex `path=/storage/*/file/*/metadata`

**Query Parameters**

- **path** (*string*) – Matches path in log lines. Default is `/`.
- **first** (*integer*) – Number of first row to return. Default is `0`.
- **rows** (*integer*) – Number of rows to return. Default is `100`. Cannot be greater than 1000.
- **starttime** (*string*) – ISO 8601 time, for lower limit of rows to return.
- **endtime** (*string*) – ISO 8601 time, for upper limit of rows to return.
- **wildcard** (*boolean*) –
  - `true` (default) - Treat end of path to have a `*` wildcard.
  - `false` - Do truncation at end of path.
- **username** (*string*) – Only return rows that the specified user invoked. Default is all rows.
- **method** (*string*) – Only return rows with the specified method, e.g. `GET`. Default is all rows.
- **performCount** (*boolean*) –
  - `true` - Return a total number of rows matching criteria (except first and count).
  - `false` (default) - Do not return a total number of rows matching criteria.
- **sort** (*string*) –
  - `asc` - Order by timestamp ascending.
  - `desc` (default) - Order by timestamp descending.

- **body** (*boolean*) –
  - `true` - Display body and response codes, if available. To activate logging of the body and response code, see the *auditTrailIncludeBody* configuration property.
  - `false` (default) - Do not display body and response codes.

**Produces**

- **application/xml, application/json** – AuditLogDocument

**Role** _administrator

### Example

```
GET /log?path=/item/VX-10&method=GET&username=admin&performCount=true
```

```
<AuditLogDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <count>13</count>
  <entry timestamp="2010-11-26T15:46:25.328+01:00">
    <username>admin</username>
    <method>GET</method>
    <path>/item/VX-10/uri</path>
    <queryParameters>methodType=AUTO</queryParameters>
    <matrixParameters/>
  </entry>
  <entry timestamp="2010-11-26T15:46:20.053+01:00">
    <username>admin</username>
    <method>GET</method>
    <path>/item/VX-10/uri</path>
    <queryParameters/>
    <matrixParameters/>
  </entry>
  <entry timestamp="2010-11-26T15:28:03.674+01:00">
    <username>admin</username>
    <method>GET</method>
    <path>/item/VX-10</path>
    <queryParameters>content=shape</queryParameters>
    <matrixParameters/>
  </entry>
  <entry timestamp="2010-11-26T15:26:49.031+01:00">
    <username>admin</username>
    <method>GET</method>
    <path>/item/VX-10</path>
    <queryParameters>content=shape</queryParameters>
    <matrixParameters/>
  </entry>
  <entry timestamp="2010-11-26T15:16:53.508+01:00">
    <username>admin</username>
    <method>GET</method>
    <path>/item/VX-10</path>
    <queryParameters>content=shape</queryParameters>
    <matrixParameters/>
  </entry>
</AuditLogDocument>
```

#### Retrieve the entire audit log

**GET /log/export**
Is very similar to the method above, but instead of delivering the entire document at once it is streamed. Therefore there is no restriction on the maximum number of rows that can be retrieved.

**Query Parameters**

- **path** (*string*) – Matches path in log lines. Default is `/`.
- **first** (*integer*) – Number of first row to return. Default is `0`.
- **rows** (*integer*) – Number of rows to return. Default is `100`.
- **starttime** (*string*) – ISO 8601 time, for lower limit of rows to return.
- **endtime** (*string*) – ISO 8601 time, for upper limit of rows to return.
- **wildcard** (*boolean*) –
  - `true` (default) - Treat end of path to have a * wildcard.
  - `false` - Do truncation at end of path.
- **username** (*string*) – Only return rows that the specified user invoked. Default is all rows.
- **method** (*string*) – Only return rows with the specified method, e.g. `GET`. Default is all rows.
- **sort** (*string*) –
  - `asc` - Order by timestamp ascending.
  - `desc` (default) - Order by timestamp descending.
- **body** (*boolean*) –
  - `true` - Display body and response codes, if available. To activate logging of the body and response code, see the *auditTrailIncludeBody* configuration property.
  - `false` (default) - Do not display body and response codes.

**Produces**

- **application/xml** – AuditLogDocument

**Role** _administrator

## 17.3 Collections

A collection is an ordered logical set of items, libraries and other collections.

**Tip:** Access to collections and their item and collection content can be set using *access control lists*.

### 17.3.1 Managing collections

#### List all collections

**GET /collection**
Retrieves a list of all known collections.

**Query Parameters**

- **first** (*integer*) – From the resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **cursor** (*string*) – New in version 4.16.
  - `*` - The initial cursor.
  - `string-from-search` - Cursor string returned from the search results.

  If set, the cursorMark (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#fetching-a-large-number-of-sorted-results-cursors) / search after (https://www.elastic.co/guide/en/elasticsearch/reference/6.8/search-request-search-after.html#search-request-search-after) features from Solr/Elasticsearch would be used to improve the deep paging (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#performance-problems-with-deep-paging) performance during a search.

  When `cursor` is used, The value of `first` will be ignored.

  Changed in version 5.5.

  Starting in 5.5, cursor is returned for the end of the result instead of null to enable tailing (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#tailing-a-cursor) search.
- **number** (*integer*) – The number of entities to fetch. Default is `100`.
- **count** (*boolean*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **content** (*string*) – Comma-separated list of additional content to retrieve. Valid values are: `metadata`, `merged-access`, `external`.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.

- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 – t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **language** (*string*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **sampleRate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **conflict** (*string*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **include** (*string*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** (*boolean*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.
- **includeValues** (*boolean*) – Return the value enumeration for each metadata field.
- **terse** (*string*) –
  - `yes` - Return metadata in *terse format*.
  - `no` (default) - Return metadata in verbose format.
- **revision** (*string*) – Specifying what revision of metadata to display. Only used if requesting a single item or collection.
- **mergedType** (*string*) – The type of operation to check for.
- **mergedPermission** (*string*) – The lowest required permission level.
- **mergedExtradata** (*string*) – Any possible extra data.

**Produces**

- **application/xml, application/json** – CollectionListDocument
- **text/plain** – CRLF-delimited list of ids

**Role** _collection_read

### Create a collection

**POST /collection**

Generates a new collection and returns the id associated with that collection.

**Query Parameters**

- **name** (*string*) – Name of the collection.
- **externalId** (*string*) – An external identifier to assign to the collection.
- **settings** (*string*) – Pre-configured *import settings*.

**Produces**

- **application/xml, application/json** – CollectionDocument
- **text/plain** – CRLF-delimited list of ids

**Status Codes**

- **400** – If the external id is already in use.

**Role** _collection_write

**POST /collection**

Generates a new collection and returns the id associated with that collection.

This resource accepts a collection document that can contain both metadata that should be set for the collection and the entities that it should contain.

**Query Parameters**

- **name** (*string*) – Name of the collection.
- **externalId** (*string*) – An external identifier to assign to the collection.
- **settings** (*string*) – Pre-configured *import settings*.

**Accepts**

- **application/xml, application/json** – CollectionDocument

**Produces**

- **application/xml, application/json** – CollectionDocument
- **text/plain** – CRLF-delimited list of ids

**Status Codes**

- **400** – If the external id is already in use.

**Role** _collection_write

### Delete a collection

**DELETE /collection/** (*collection-id*)

Delete specified collection.

Note that the actual items and libraries that are contained within the collection are not modified.

**Status Codes**

- **200 OK** – The collection is deleted.

**Role** _collection_write

#### Delete multiple collections

**DELETE /collection**

Delete multiple collections.

Note that the actual items and libraries that are contained within the collection are not modified.

**Query Parameters**

- **id** (*string*) – Required. Comma-separated list of collection ids or external ids.

**Status Codes**

- **200 OK** – The collections are deleted.
- **404 Not found** – Could not find the collection.

**Role** _collection_write

#### Example

```
DELETE /collection?id=VX-32,VX-56
```

#### Update collection name

**PUT /collection/** (*collection-id*) **/rename**

Sets the name of the collection with the *Identifiers* `collection-id`.

**Query Parameters**

- **name** (*string*) – Required. New name of the collection.

**Role** _collection_write

### 17.3.2 Collection content

#### Retrieve a collection

**GET /collection/** (*collection-id*)

Return the ids of the objects contained within the collection, that has the id `collection-id`.

**Query Parameters**

- **content** (*string*) – Comma-separated list of additional content to retrieve. Valid values are: `metadata`, `merged-access`, `external`.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.

  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* : *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1* − *t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* * - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **language** (*string*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **sampleRate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **conflict** (*string*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **include** (*string*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** (*boolean*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.
- **includeValues** (*boolean*) – Return the value enumeration for each metadata field.
- **terse** (*string*) –
  - `yes` - Return metadata in *terse format*.
  - `no` (default) - Return metadata in verbose format.

- **revision** (*string*) – Specifying what revision of metadata to display. Only used if requesting a single item or collection.
- **mergedType** (*string*) – The type of operation to check for.
- **mergedPermission** (*string*) – The lowest required permission level.
- **mergedExtradata** (*string*) – Any possible extra data.
- **children** (*string*) – Comma-separated list of types to include in the result. Default is to return everything.
  - `collection` - Return collections contained in this collection.
  - `item` - Return items contained in this collection.
  - `library` - Return libraries contained in this collection.

  New in version 4.16.6.

**Status Codes**

- **404 Not found** – Could not find the collection.

**Produces**

- **application/xml, application/json** – CollectionDocument
- **text/plain** – CRLF-delimited list of ids

**Role** _collection_read

### Retrieve the items of a collection

**GET /collection/** (*collection-id*) **/item**

Retrieves only the items of the collection.

Queries on collection items will now return items in creation order by default. See *indexCollectionItemOrder* on how to revert back to using the insert/custom collection item ordering.

**Content Parameters** See *Retrieving item information*

**Query Parameters**

- **result** (*string*) –
  - `list` (default) - Return a list of items.
  - `library` - Create a library with the matching items.
- **q** (*string*) – XML/JSON, ItemSearchDocument. Only with GET (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3).
- **library** (*string*) – Restricts search to within library, *Identifiers*. Default is `*`, all items.
- **first** (*integer*) – From the resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **cursor** (*string*) – New in version 4.16.
  - `*` - The initial cursor.
  - `string-from-search` - Cursor string returned from the search results.

If set, the cursorMark (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#fetching-a-large-number-of-sorted-results-cursors) / search after (https://www.elastic.co/guide/en/elasticsearch/reference/6.8/search-request-search-after.html#search-request-search-after) features from Solr/Elasticsearch would be used to improve the deep paging (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#performance-problems-with-deep-paging) performance during a search.

When `cursor` is used, The value of `first` will be ignored.

Changed in version 5.5.

Starting in 5.5, cursor is returned for the end of the result instead of null to enable tailing (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#tailing-a-cursor) search.

- **number** (*integer*) – The number of entities to fetch. Default is `100`.
- **count** (*boolean*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **libraryId** (*string*) – If set, the library identified by this id will be used instead of creating a new library.
- **autoRefresh** (*boolean*) – When creating a library, make it *self-refresh*. Default is `false`.
- **updateMode** (*string*) – When creating a library, use this *update mode*. Default is `MERGE`.
- **updateFrequency** (*string*) – When creating a library, use this *update frequency*. Defaults to no periodic updates.
- **p** (*string*) – Comma-separated list of *paths* specifying the content to include. Overrides the content and filter parameters.
- **content** (*string*) – Comma-separated list of the types of content to retrieve. Valid values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.

  - *group-name* + - Return specified group and subgroups.
  - *group-name* `:` *new-name* - Return specified group, renamed to a new name in return value.
  - `-` *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **`language`** (*`string`*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **`sampleRate`** (*`string`*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **`track`** (*`string`*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1* `-` *t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **`include`** (*`string`*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **`includeValues`** (*`boolean`*) – Return the value enumeration for each metadata field.
- **`conflict`** (*`string`*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **`terse`** (*`string`*) –
  - `yes` - Return metadata in *terse format*.
  - `no` (default) - Return metadata in verbose format.
- **`defaultValue`** (*`boolean`*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **`includeTransientMetadata`** (*`boolean`*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.
- **`revision`** (*`string`*) – Specifying which metadata revision to display. Only used if requesting a single item or collection.
- **`mergedType`** (*`string`*) – The type of operation to check for.
- **`mergedPermission`** (*`string`*) – The lowest required permission level.

- **mergedExtradata** (*string*) – Any possible extra data.
- **uriType** (*string*) – Comma-separated list of format types (container format) to return.
- **scheme** (*string*) – URI scheme to return.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodType** (*string*) – *Access method*.
  - `AUTO` - Gives an APInoauth URI to the media. Access to file is tunneled through Vidispine.
  - `AZURE_SAS` - If the storage schema is azure:// you can get direct access to the media. The resulting URI will not tunnel through Vidispine but rather point directly to the media location at the azure storage.
- **methodMetadata** (*string*) – *Metadata* used with storage method.
- **tag** (*string*) – A *URI parameter*: Comma-separated list of *shape tags* to return.
- **version** (*string*) – Specifying which essence version to return for shapes. If special value `all`, display all versions. If special value `latest` (default), display latest version of shapes.
- **closedFiles** (*boolean*) – A *URI parameter*:
  - `true` (default) - Return only URIs that point to closed files.
  - `false` - Return all URIs.
- **storage** (*string[]*) – List of storage ids. Return only files from specific storages. Can be specified multiple times.
- **storageGroup** (*string*) – Storage group id. Return only files from storages specified in the storage group.
- **starttc** (*boolean*) –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.
- **url** (*boolean*) –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.
- **noauth-url** (*boolean*) –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **baseURI** (*string*) – Which base URI to use for the thumbnail URLs.
- **save** (*boolean*) –
  - `true` - Returns a `303 See Other`, with a `Location` header containing an URI to fetch the search result
  - `false` (default) - Returns a regular search result

**Produces**

- **application/xml, application/json** – ItemListDocument
- **text/plain** – CRLF-delimited list of ids or URLs

- **application/xml, application/json** – ItemListDocument
- **text/plain** – CRLF-delimited list of ids or URLs.

**Status Codes**

- **404 Not found** – Could not find the collection.

**Role** _collection_read

**Role** _metadata_read (`content=metadata`)

**Role** _item_uri (`content=uri`)

**Role** _thumbnail_read (`content=poster` and `content=thumbnail`)

**Role** _access_control_read (`content=access` and `content=merged-access`)

**Role** _item_id_read (`content=external`)

### Create an item list job for the collection

**POST /collection/** (*collection-id*) **/item/list**

Starts a new job that creates a list of all items in the collection, similarly to *listing all items*.

**Query Parameters**

- **recursive** (*boolean*) – will include items in child collections recursively. Default is false.
- **destinationUri** (*string*) – Required. The URI to output the CSV file to.
- **username** (*string*) – Filter items according to the access of the specified user.
- **field** (*string*) – Comma-separated list of metadata fields to include in the result. Default is `title`
- **outputFormat** (*string*) – Specifies the output format. One of `xml` (default) and `csv`.
- **data** (*string*) – Specifies any additional data that should be included with the metadata fields.
- **p** (*string*) – Comma-separated list of *paths* specifying the content to include. Overrides the field and data parameters. Only supported for XML output.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Accepts**

- **application/xslt** – An optional XSLT capable of transforming ItemListDocument.

**Produces**

- **application/xml, application/json** – JobDocument.

**Role** _administrator

### Retrieve the ancestors of a collection

**GET /collection/** (*collection-id*) **/ancestor**
Retrieves the ids of all ancestors of the collection.

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – CRLF-delimited list of ids

**Role** _collection_read

### Search for items within a collection

**PUT /collection/** (*collection-id*) **/item**
Performs a search among the items in the specified collection.

---

**Note:** Searching can also be performed by using the HTTP method PUT (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.6) using the same syntax, except for the parameter `q` is omitted and the ItemSearchDocument is sent in the body of the request.

---

**Tip:** There is a limit on how many items that can be returned for each call to this method. To get all items, iterate the calls, or even better in a batch scenario, use *Listing items in batch*.

---

Queries on collection items will now return items in creation order by default. See *indexCollectionItemOrder* on how to revert back to using the insert/custom collection item ordering.

**Content Parameters** See *Retrieving item information*

**Query Parameters**

- **result** (*string*) –
  - `list` (default) - Return a list of items.
  - `library` - Create a library with the matching items.
- **q** (*string*) – XML/JSON, ItemSearchDocument. Only with GET (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3).
- **library** (*string*) – Restricts search to within library, *Identifiers*. Default is `*`, all items.
- **first** (*integer*) – From the resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **cursor** (*string*) – New in version 4.16.
  - `*` - The initial cursor.
  - `string-from-search` - Cursor string returned from the search results.

  If set, the cursorMark (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#fetching-a-large-number-of-sorted-results-cursors) / search after (https://www.elastic.co/guide/en/elasticsearch/reference/6.8/search-request-search-after.html#search-request-search-after) features from Solr/Elasticsearch would be used to improve the deep paging (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#performance-problems-with-deep-paging) performance during a search.

  When `cursor` is used, The value of `first` will be ignored.

Changed in version 5.5.

Starting in 5.5, cursor is returned for the end of the result instead of null to enable tailing (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#tailing-a-cursor) search.

- **number** (*integer*) – The number of entities to fetch. Default is `100`.
- **count** (*boolean*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **libraryId** (*string*) – If set, the library identified by this id will be used instead of creating a new library.
- **autoRefresh** (*boolean*) – When creating a library, make it *self-refresh*. Default is `false`.
- **updateMode** (*string*) – When creating a library, use this *update mode*. Default is `MERGE`.
- **updateFrequency** (*string*) – When creating a library, use this *update frequency*. Defaults to no periodic updates.
- **p** (*string*) – Comma-separated list of *paths* specifying the content to include. Overrides the content and filter parameters.
- **content** (*string*) – Comma-separated list of the types of content to retrieve. Valid values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **language** (*string*) – Comma-separated list.

  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **sampleRate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 – t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **include** (*string*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **includeValues** (*boolean*) – Return the value enumeration for each metadata field.
- **conflict** (*string*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **terse** (*string*) –
  - `yes` - Return metadata in *terse format*.
  - `no` (default) - Return metadata in verbose format.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** (*boolean*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.
- **revision** (*string*) – Specifying which metadata revision to display. Only used if requesting a single item or collection.
- **mergedType** (*string*) – The type of operation to check for.
- **mergedPermission** (*string*) – The lowest required permission level.
- **mergedExtradata** (*string*) – Any possible extra data.
- **uriType** (*string*) – Comma-separated list of format types (container format) to return.
- **scheme** (*string*) – URI scheme to return.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodType** (*string*) – *Access method*.

  - `AUTO` - Gives an APInoauth URI to the media. Access to file is tunneled through Vidispine.
  - `AZURE_SAS` - If the storage schema is azure:// you can get direct access to the media. The resulting URI will not tunnel through Vidispine but rather point directly to the media location at the azure storage.
- **methodMetadata** (*string*) – *Metadata* used with storage method.
- **tag** (*string*) – A *URI parameter*: Comma-separated list of *shape tags* to return.
- **version** (*string*) – Specifying which essence version to return for shapes. If special value `all`, display all versions. If special value `latest` (default), display latest version of shapes.
- **closedFiles** (*boolean*) – A *URI parameter*:
  - `true` (default) - Return only URIs that point to closed files.
  - `false` - Return all URIs.
- **storage** (*string[]*) – List of storage ids. Return only files from specific storages. Can be specified multiple times.
- **storageGroup** (*string*) – Storage group id. Return only files from storages specified in the storage group.
- **starttc** (*boolean*) –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.
- **url** (*boolean*) –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.
- **noauth-url** (*boolean*) –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **baseURI** (*string*) – Which base URI to use for the thumbnail URLs.
- **save** (*boolean*) –
  - `true` - Returns a `303 See Other`, with a `Location` header containing an URI to fetch the search result
  - `false` (default) - Returns a regular search result

**Produces**

- **application/xml, application/json** – ItemListDocument
- **text/plain** – CRLF-delimited list of ids or URLs

**Accepts**

- **application/xml, application/json** – ItemSearchDocument

**Status Codes**

- **400 Bad request** – Either the ItemSearchDocument or a parameter was invalid.

**Role** _collection_read

**Role** _metadata_read (`content=metadata`)

**Role** _item_uri (`content=uri`)

**Role** _thumbnail_read (`content=poster` and `content=thumbnail`)

**Role** _access_control_read (`content=access` and `content=merged-access`)

**Role** _item_id_read (`content=external`)

### Example

```
GET /collection/VX-76/item
Accept: application/xml
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <item id="VX-45"/>
  <item id="VX-46"/>
  <item id="VX-47"/>
  <item id="VX-62"/>
</ItemListDocument>
```

### Add an item, library or collection to a collection

**PUT /collection/** (*collection-id*) **/**
*id* Adds an item, library or collection with the id `id`, to the collection with the id `collection-id`. If `id` is already present within the collection, this is a no-op, except if the query parameter metadata is used. In that case, metadata is updated for the specified entity.

**Query Parameters**

- **type** (*string*) –
  - `collection` - The object identified by `id` is a collection.
  - `item` (default) - The object identified by `id` is an item.
  - `library` - The object identified by `id` is a library.
- **reference** (*string*) – an optional UUID of the relation between the collection and the added entity
- **mode** (*string*) –
  - `REPLACE` (default) - Existing relations between the collection and the added entity are removed before entity is added
  - `ADD` - Entity is added to the list of relation, the entity can appear multiple times
- **before** (*string*) – an optional UUID of where in the list of relations the new relation should be inserted. The relation is inserted before the relation given. If no UUID is give, the relation is added at the end.
- **addItems** (*boolean*) –
  - `true` - Library items will be added individually. Only has any effect when `type=library`.
  - `false` - Library will be added to collection, not specific items.
- **metadata** (*string[]*) –

- *key* = *value* - Set or add metadata field to the relation between the collection and entity. Can be used multiple times to add several fields. The key cannot be empty or start with a minus sign. To delete a field enter the same key with a minus sign at the beginning.

Note that = is part of the query parameter, and has to be encoded (`%3d`).

**Status Codes**

- **`200 OK`** – The collection, item or library was added, or already existed within the collection.
- **`400 Bad request`** – Cannot add a collection to itself, the type was given an invalid value or a metadata key is not specified.
- **`404 Not found`** – Could not find the collection, item or library.

**Role** _collection_write

### Example

```
PUT /collection/VX-1000/VX*10?type=library&metadata=addedBy%3dadmin&metadata=dateAdded%3
```

```
HTTP/1.1 200 OK
```

### Remove an item, library or collection from a collection

**`DELETE /collection/`**(*collection-id*)**`/`**

*id* Attempts to remove specific content with the id, `id`, from a collection with the id `collection-id`.

Note that the object corresponding to the id is not altered.

**Query Parameters**

- **`type`** (*`string`*) –
  - `collection` - The object identified by `id` is a collection.
  - `item` (default) - The object identified by `id` is an item.
  - `library` - The object identified by `id` is a library.
- **`reference`** (*`string`*) – an optional UUID of the relation to be deleted. Can be used to delete a specific instance of relation between the collection and the referenced entity.

**Status Codes**

- **`200 OK`** – The item/library is removed from the collection.
- **`400 Bad request`** – The type was given an invalid value.
- **`404 Not found`** – Could not find the collection or the item/library.

**Role** _collection_write

### Update a collection

**`PUT /collection/`**(*collection-id*)

Updates the content of the collection with the id `collection-id` as specified in the document. It is also possible to change the name of the collection and metadata of the collection-entity relations.

Either all or no entities must have a mode specified. If no entities have a mode specified and the document contains an entity that does not exist in the collection, then the entity will be added.

When no entities have a mode specified the entities will get the same position as they are ordered in the document.

**Query Parameters**

- **clear** (*boolean*) –
  - `true` (default) - All entities that are in the collection but not specified in the document will be removed. Only has any effect when no entities have a mode specified.
  - `false` - All entities in the document will be appended to the collection. If an entity already exist in the collection then the position is determined by the document. Only has any effect when no entities have a mode specified.

**Accepts**

- **application/xml, application/json** – CollectionDocument that contains the entity ids.

**Produces**

- **application/xml, application/json** – CollectionDocument containing the collection name and the entities in order.

**Status Codes**

- **200 OK** – All operations in the document was successfully completed.
- **404 Not found** – Could not find one of the entities.
- **400 Invalid input** – Invalid type specified, all or none entities have not a mode specified or an entity is specified twice in the document.

**Role** _collection_write

### Example: Modify an entire collection

Here is an example where the name of the collection is changed and some entities are added, removed and repositioned. First *GET /collection/(collection-id)* is used to get the content. The content is then modified and applied with `PUT /collection/(collection-id)?clear=true`.

```
GET /collection/VX-1000
```

```
<CollectionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <loc>http://localhost:8080/API/collection/VX-1000/</loc>
  <id>VX-1000</id>
  <name>Old name</name>
  <content>
    <id>VX-1</id>
    <uri>http://localhost:8080/API/item/VX-1</uri>
    <type>item</type>
    <metadata/>
  </content>
  <content>
    <id>VX-2</id>
    <uri>http://localhost:8080/API/item/VX-2</uri>
    <type>item</type>
    <metadata/>
  </content>
  <content>
    <id>VX-3</id>
    <uri>http://localhost:8080/API/item/VX-3</uri>
```

```
    <type>item</type>
    <metadata>
      <field>
        <key>AddedBy</key>
        <value>foo</value>
      </field>
    </metadata>
  </content>
  <content>
    <id>VX*20</id>
    <uri>http://localhost:8080/API/library/VX*20</uri>
    <type>library</type>
    <metadata/>
  </content>
  <content>
    <id>VX-100</id>
    <uri>http://localhost:8080/API/collection/VX-100</uri>
    <type>collection</type>
    <metadata>
      <field>
        <key>AddedBy</key>
        <value>Admin</value>
      </field>
    </metadata>
  </content>
  <content>
    <id>VX-200</id>
    <uri>http://localhost:8080/API/collection/VX-200</uri>
    <type>collection</type>
    <metadata/>
  </content>
  <content>
    <id>VX-300</id>
    <uri>http://localhost:8080/API/collection/VX-300</uri>
    <type>collection</type>
    <metadata/>
  </content>
</CollectionDocument>
```

```
PUT /collection/VX-1000?clear=true
Content-Type: application/xml

<CollectionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <loc>http://localhost:8080/API/collection/VX-1000/</loc>
  <id>VX-1000</id>
  <!-- Change the name to "New name" -->
  <name>New name</name>

  <!-- Swap the positions of item VX-1 and item VX-2 -->
  <content>
    <id>VX-2</id>
    <uri>http://localhost:8080/API/item/VX-2</uri>
    <type>item</type>
    <metadata/>
  </content>
  <content>
    <id>VX-1</id>
    <uri>http://localhost:8080/API/item/VX-1</uri>
```

```
    <type>item</type>
    <metadata/>
  </content>

  <!-- Change value on the key "AddedBy" to "bar" in the metadata field of item VX-3 -
↪->
  <content>
    <id>VX-3</id>
    <uri>http://localhost:8080/API/item/VX-3</uri>
    <type>item</type>
    <metadata>
      <field>
        <key>AddedBy</key>
        <value>bar</value>
      </field>
    </metadata>
  </content>

  <!-- Add library V*10. If addItems=true then the items in the library
  will be added instead of the library. addItems is by default false
  and only has effect on libraries. -->

  <content addItems="false">
    <id>VX*10</id>
    <uri>http://localhost:8080/API/library/VX*10</uri>
    <type>library</type>
    <metadata/>
  </content>
  <content>
    <id>VX*20</id>
    <uri>http://localhost:8080/API/library/VX*20</uri>
    <type>library</type>
    <metadata/>
  </content>

<!-- Remove the metadata from VX-100 -->
  <content>
    <id>VX-100</id>
    <uri>http://localhost:8080/API/collection/VX-100</uri>
    <type>collection</type>
  </content>

  <content>
    <id>VX-200</id>
    <uri>http://localhost:8080/API/collection/VX-200</uri>
    <type>collection</type>
    <metadata/>
  </content>

  <!-- Collection VX-300 will be removed since clear=true and the
  collection VX-300 is not specified here in the document -->
</CollectionDocument>
```

```
<CollectionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <content>
        <id>VX-2</id>
        <type>item</type>
        <uri>http://localhost:8080/API/item/VX-2</uri>
```

```
        <metadata/>
    </content>
    <content>
        <id>VX-1</id>
        <type>item</type>
        <uri>http://localhost:8080/API/item/VX-1</uri>
        <metadata/>
    </content>
    <content>
        <id>VX-3</id>
        <type>item</type>
        <uri>http://localhost:8080/API/item/VX-3</uri>
        <metadata>
          <field>
            <key>AddedBy</key>
            <value>bar</value>
          </field>
        </metadata>
    </content>
    <content>
        <id>VX*10</id>
        <type>library</type>
        <uri>http://localhost:8080/API/library/VX*10</uri>
        <metadata/>
    </content>
    <content>
        <id>VX*20</id>
        <type>library</type>
        <uri>http://localhost:8080/API/library/VX*20</uri>
        <metadata/>
    </content>
    <content>
        <id>VX-100</id>
        <type>collection</type>
        <uri>http://localhost:8080/API/collection/VX-100</uri>
    </content>
    <content>
        <id>VX-200</id>
        <type>collection</type>
        <uri>http://localhost:8080/API/collection/VX-200</uri>
        <metadata/>
    </content>
    <id>VX-1000</id>
    <name>New name</name>
</CollectionDocument>
```

### Example: Specify each change of a collection

To state each change, instead of using a document with all entities, a content mode can be specified to each content. The different modes are “add”, “remove” and “move”. The following example is a continuation of the previous example.

```
PUT /collection/VX-1000
Content-Type: application/xml

<CollectionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <loc>http://localhost:8080/API/collection/VX-1000/</loc>
  <id>VX-1000</id>
```

```
  <!-- Change the name of the collection to "Final name" -->
  <name>Final name</name>

  <!-- Remove item VX-3 -->
  <content mode="remove">
    <id>VX-3</id>
    <type>item</type>
  </content>

  <!-- Add library VX*30 after library VX*20 with metadata-->
  <content mode="add" after="VX*20">
    <id>VX*30</id>
    <type>library</type>
    <metadata>
        <field>
            <key>AddedBy</key>
            <value>User</value>
        </field>
    </metadata>
  </content>

  <!-- Add the all items from library VX*10 (which are VX-97 and VX-98)
  after item VX-2 -->
  <content mode="add" addItems="true" after="VX-2">
    <id>VX*10</id>
    <type>library</type>
  </content>

  <!-- Move library VX*20 before VX*10, metadata won't be updated for specified␣
↪entity when mode="move" -->
  <content mode="move" before=VX*10>
    <id>VX*20</id>
    <type>library</type>
  </content>

  <!-- Note that first we added library VX*30 after library VX*20 then
  we moved library VX*20 before VX*10:
  [VX*10, VX*20] -> [VX*10, VX*20, VX*30] -> [VX*20, VX*10, VX*30] -->
</CollectionDocument>
```

```
<CollectionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <id>VX-1000</id>
    <name>Final name</name>
    <content>
        <id>VX-2</id>
        <type>item</type>
        <uri>http://localhost:8080/API/item/VX-2</uri>
    </content>
    <content>
        <id>VX-97</id>
        <type>item</type>
        <uri>http://localhost:8080/API/item/VX-97</uri>
    </content>
    <content>
        <id>VX-98</id>
        <type>item</type>
        <uri>http://localhost:8080/API/item/VX-98</uri>
    </content>
```

```
    <content>
        <id>VX-1</id>
        <type>item</type>
        <uri>http://localhost:8080/API/item/VX-1</uri>
    </content>
    <content>
        <id>VX*20</id>
        <type>library</type>
        <uri>http://localhost:8080/API/library/VX*20</uri>
    </content>
    <content>
        <id>VX*10</id>
        <type>library</type>
        <uri>http://localhost:8080/API/library/VX*10</uri>
    </content>
    <content>
        <id>VX*30</id>
        <type>library</type>
        <uri>http://localhost:8080/API/library/VX*30</uri>
        <metadata>
            <field>
                <key>AddedBy</key>
                <value>User</value>
            </field>
        </metadata>
    </content>
    <content>
        <id>VX-100</id>
        <type>collection</type>
        <uri>http://localhost:8080/API/collection/VX-100</uri>
    </content>
    <content>
        <id>VX-200</id>
        <type>collection</type>
        <uri>http://localhost:8080/API/collection/VX-200</uri>
    </content>
</CollectionDocument>
```

### 17.3.3 Collection metadata

Metadata can be set on collections, just like with items.

See *Metadata*.

### 17.3.4 Searching for collections

Searching collections behaves much like *Search*.

#### Search for collections

**PUT /collection**

Searches for collections that matches the query.

**Query Parameters**

- **first** (*integer*) – The index of the first collection. Default is `1`.
- **cursor** (*string*) – New in version 4.16.

  - `*` - The initial cursor.
  - `string-from-search` - Cursor string returned from the search results.

  If set, the cursorMark (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#fetching-a-large-number-of-sorted-results-cursors) / search after (https://www.elastic.co/guide/en/elasticsearch/reference/6.8/search-request-search-after.html#search-request-search-after) features from Solr/Elasticsearch would be used to improve the deep paging (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#performance-problems-with-deep-paging) performance during a search.

  When `cursor` is used, The value of `first` will be ignored.

  Changed in version 5.5.

  Starting in 5.5, cursor is returned for the end of the result instead of null to enable tailing (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#tailing-a-cursor) search.

- **number** (*integer*) – The number of collections to retrieve. Default is `100`.
- **count** (*boolean*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **content** (*string*) – Comma-separated list of additional content to retrieve. Valid values are: `metadata`, `merged-access`, `external`.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF--+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.

  - *track-type t1 – t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **`language`** (*`string`*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **`sampleRate`** (*`string`*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **`conflict`** (*`string`*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **`include`** (*`string`*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **`defaultValue`** (*`boolean`*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **`includeTransientMetadata`** (*`boolean`*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.
- **`includeValues`** (*`boolean`*) – Return the value enumeration for each metadata field.
- **`terse`** (*`string`*) –
  - `yes` - Return metadata in *terse format*.
  - `no` (default) - Return metadata in verbose format.
- **`revision`** (*`string`*) – Specifying what revision of metadata to display. Only used if requesting a single item or collection.
- **`mergedType`** (*`string`*) – The type of operation to check for.
- **`mergedPermission`** (*`string`*) – The lowest required permission level.
- **`mergedExtradata`** (*`string`*) – Any possible extra data.

**Accepts**

- **`application/xml, application/json`** – ItemSearchDocument

**Produces**

- **`application/xml, application/json`** – CollectionListDocument

**Role** _collection_read

### Retrieve the search history

**GET /collection/history**

Retrieves a list of searches made by a particular user, include "Collection search " and "Item and collection search". The results are ordered according to timestamp, with the latest searches being first. Duplicate queries will not be retrieved.

**Query Parameters**

- **start** (*string*) – If set, only searches made after this date will be retrieved.
- **maxResults** (*integer*) – The maximum number of searches that will be retrieved. The value must be between 1 and 50. Default is `10`.
- **username** (*string*) – The name of the user that has performed the searched. If not specified, the user performing the request will be selected.

**Status Codes**

- **400 Bad request** – The request was malformed.

**Produces**

- **application/xml, application/json** – SearchHistoryDocument

**Role** _item_search

### Retrieve the child-collections of a collection

**GET /collection/** (*collection-id*) **/collection**

Returns the child collections of collection with id `collection-id`.

**Query Parameters**

- **first** (*integer*) – From the resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **cursor** (*string*) – New in version 4.16.
  - `*` - The initial cursor.
  - `string-from-search` - Cursor string returned from the search results.

  If set, the cursorMark (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#fetching-a-large-number-of-sorted-results-cursors) / search after (https://www.elastic.co/guide/en/elasticsearch/reference/6.8/search-request-search-after.html#search-request-search-after) features from Solr/Elasticsearch would be used to improve the deep paging (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#performance-problems-with-deep-paging) performance during a search.

  When `cursor` is used, The value of `first` will be ignored.

  Changed in version 5.5.

  Starting in 5.5, cursor is returned for the end of the result instead of null to enable tailing (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#tailing-a-cursor) search.
- **number** (*integer*) – The number of entities to fetch. Default is `100`.
- **count** (*boolean*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.

- **content** (*string*) – Comma-separated list of additional content to retrieve. Valid values are: `metadata`, `merged-access`, `external`.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* `:` *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 − t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **language** (*string*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **sampleRate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **conflict** (*string*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.

- **include** (*string*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** (*boolean*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.
- **includeValues** (*boolean*) – Return the value enumeration for each metadata field.
- **terse** (*string*) –
  - `yes` - Return metadata in *terse format*.
  - `no` (default) - Return metadata in verbose format.
- **revision** (*string*) – Specifying what revision of metadata to display. Only used if requesting a single item or collection.
- **mergedType** (*string*) – The type of operation to check for.
- **mergedPermission** (*string*) – The lowest required permission level.
- **mergedExtradata** (*string*) – Any possible extra data.

**Status Codes**

- **404 Not found** – Could not find the collection.

**Produces**

- **application/xml, application/json** – CollectionListDocument
- **text/plain** – CRLF-delimited list of ids

**Role** _collection_read

### Search the child-collections of a collection

**PUT /collection/** (*collection-id*) **/collection**

Searches for collections that matches the query, limited to the child-collections of `collection-id`.

**Query Parameters**

- **first** (*integer*) – The index of the first collection. Default is `1`.
- **cursor** (*string*) – New in version 4.16.
  - `*` - The initial cursor.
  - `string-from-search` - Cursor string returned from the search results.

  If set, the cursorMark (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#fetching-a-large-number-of-sorted-results-cursors) / search after (https://www.elastic.co/guide/en/elasticsearch/reference/6.8/search-request-search-after.html#search-request-search-after) features from Solr/Elasticsearch would be used to improve the deep paging (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#performance-problems-with-deep-paging) performance during a search.

  When `cursor` is used, The value of `first` will be ignored.

Changed in version 5.5.

Starting in 5.5, cursor is returned for the end of the result instead of null to enable tailing (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#tailing-a-cursor) search.

- **number** (*integer*) – The number of collections to retrieve. Default is `100`.
- **count** (*boolean*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **content** (*string*) – Comma-separated list of additional content to retrieve. Valid values are: `metadata`, `merged-access`, `external`.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 − t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* ⋆ - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **language** (*string*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.

  - `all` (default) - Return all metadata, with or without language specification.
- **sampleRate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **conflict** (*string*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **include** (*string*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** (*boolean*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.
- **includeValues** (*boolean*) – Return the value enumeration for each metadata field.
- **terse** (*string*) –
  - `yes` - Return metadata in *terse format*.
  - `no` (default) - Return metadata in verbose format.
- **revision** (*string*) – Specifying what revision of metadata to display. Only used if requesting a single item or collection.
- **mergedType** (*string*) – The type of operation to check for.
- **mergedPermission** (*string*) – The lowest required permission level.
- **mergedExtradata** (*string*) – Any possible extra data.

**Accepts**

- **application/xml, application/json** – ItemSearchDocument

**Produces**

- **application/xml, application/json** – CollectionListDocument

**Role** _collection_read

### 17.3.5 Ordering collections

Collections will return their elements in the same order for every request.

Queries on collection items will now return items in creation order by default. See *indexCollectionItemOrder* on how to revert back to using the insert/custom collection item ordering.

#### Reorder collection elements

**POST /collection/** (*collection-id*) **/order**

Changes the order of the elements. Note that the reordering elements are parsed and applied in the sequence that they are supplied.

**Accepts**

- **application/xml, application/json** – CollectionReorderDocument containing the changes to the order.

**Produces**

- **application/xml, application/json** – CollectionDocument containing the elements in their new order.
- **text/plain** – CRLF-delimited list of ids

**Role** _collection_write

### Example

Starting with an unordered collection of items, we will sort it according to item id. At the start it contains items [VX-7, VX-8, VX-5, VX-6].

```
GET /collection/VX-1
```

```
<CollectionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <loc>http://localhost:8080/API/collection/VX-1/VX-1</loc>
  <id>VX-1</id>
  <content>
    <id>VX-7</id>
    <uri>http://localhost:8080/API/item/VX-7</uri>
    <type>item</type>
  </content>
  <content>
    <id>VX-8</id>
    <uri>http://localhost:8080/API/item/VX-8</uri>
    <type>item</type>
  </content>
  <content>
    <id>VX-5</id>
    <uri>http://localhost:8080/API/item/VX-5</uri>
    <type>item</type>
  </content>
  <content>
    <id>VX-6</id>
    <uri>http://localhost:8080/API/item/VX-6</uri>
    <type>item</type>
  </content>
</CollectionDocument>
```

```
POST /collection/VX-1/order
Content-Type: application/xml

<CollectionReorderDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <!-- Find the current first element and put VX-5 first -->
    <item id="VX-5" before="VX-7"/>

    <!-- Add the other elements after VX-5 in sequence -->
    <item id="VX-6" after="VX-5"/>
    <item id="VX-7" after="VX-6"/>
    <item id="VX-8" after="VX-7"/>
</CollectionReorderDocument>
```

```
<CollectionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-1</id>
  <content>
    <id>VX-5</id>
    <uri>http://localhost:8080/API/item/VX-5</uri>
    <type>item</type>
  </content>
  <content>
    <id>VX-6</id>
    <uri>http://localhost:8080/API/item/VX-6</uri>
    <type>item</type>
  </content>
  <content>
    <id>VX-7</id>
    <uri>http://localhost:8080/API/item/VX-7</uri>
    <type>item</type>
  </content>
  <content>
    <id>VX-8</id>
    <uri>http://localhost:8080/API/item/VX-8</uri>
    <type>item</type>
  </content>
</CollectionDocument>
```

### 17.3.6 Folder mapped collections

It is possible to map a Vidispine collection to a folder on the file system. This means that any files of items part of the collection will be stored in a sub-folder with the same name as the collection. For a collection marked as mapped to a folder, some additional rules are enforced when it comes to collection relationships:

- A folder mapped collection can have at most one folder mapped parent collection.
- An item can have at most one folder mapped parent collection.

That is, the same rule that applies to files on a traditional file system.

---

**Note:** Adding an item to a folder mapped collection will not move the item files to the corresponding folder immediately as the file movement is done asynchronously in the background.

---

#### Mark a collection as folder mapped

**PUT /collection/** (*collection-id*) **/map-to-folder**
Marks collection `collection-id` as mapped to folder. Files in child items will be moved to the corresponding folder in the storages.

**Role** _collection_write

#### Unmark a collection as folder mapped

**DELETE /collection/** (*collection-id*) **/map-to-folder**
Marks collection `collection-id` as *not* mapped to folder. Files in child items will be moved to the root directory in the storages.

**Role** _collection_write

### Report that the folder name has changed on disk

**PUT /collection/** (*collection-id*) **/folder-name**

If the folder name has been changed by a user or an external program, it can be reported to Vidispine with this command. The affected file entities in the database will then be updated with the new path, and the collection name will be changed.

**Query Parameters**

- **name** (*string*) – Required. The new name of the folder.

**Produces**

- **text/plain** – "OK"

**Role** _collection_write

### Training datasets

**POST /collection/** (*collection-id*) **/train**

Train a collection to a model using a VidiNet Cognitive Service. The model can then be used for running analyses.

New in version 21.3.

**Query Parameters**

- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM`.
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **resourceId** (*string*) – Required. The VidiNet resource to use for training.
- **callbackId** (*string*) – Required. The callback resource id to use for finding and running callback scripts.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _collection_write

## 17.4 Deletion locks

Manage deletion locks.

New in version 4.15.

In the following reference, `{lock-entity}` is one of the following:

- `/collection`
- `/collection/(collection-id)`
- `/item`
- `/item/(item-id)`
- `{storage-resource}/file`
- `{file-resource}`

### 17.4.1 Manage deletion locks

#### List all locks

**GET /deletion-lock**
Retrieves a list of deletion locks.

**Query Parameters**

- **onlyEffective** (*boolean*) –
  - `true` - Only return the effective lock of the entity.
  - `false` (default) - Return all deletion locks applied on the entity.
- **first** (*integer*) – Return locks starting from the specified offset. Default is `1`, the first lock.
- **number** (*integer*) – Return at most the specified number of locks. Default is `100`.
- **metadata** (*string[]*) – Filter out only the locks that has metadata according to the filter criteria.
  - *key* = *value* - Multiple query parameters can be specified.

  Note that = is part of the query parameter, and has to be encoded (`%3d`).
- **username** (*string*) – Comma-separated user names. Filter only locks created by the specified user(s).
- **range** (*string*) – Filter out locks whose expiry time is within the specified range. The range format is `[d..d]`, `(d..d)`, `[d..d)`, `(d..d]`, `(*..d]`, `[d..*)`, or `(*..*)`. `d` is a date and time in the ISO 8601 format.
- **entityTypes** (*string*) – Comma-separated list. Only return locks set explicitly on the specified entity type(s).

  Valid values are: `item`, `collection`, `file`, and `all` (default).

**Produces**

- **application/xml, application/json** – DeletionLockListDocument

**Role** _deletion_lock_read

```
GET /deletion-lock?user=testuser
            &metadata=workflow=production
            &metadata=group=movie
            &expiry=(*..2020-10-05T16:42:34.693%2B02:00]
```

#### Retrieve a lock

**GET /deletion-lock/** (*lock-id*)
Returns a specific lock.

**Produces**

- **application/xml, application/json** – XML/JSON, schema DeletionLock-Document

**Role** _deletion_lock_read

#### List all locks for an entity

**GET {lock-entity}/deletion-lock**
Retrieves a list of deletion locks on the entity.

**Query Parameters**

- **onlyEffective** (*boolean*) –
  - `true` - Only return the effective lock of the entity.
  - `false` (default) - Return all deletion locks applied on the entity.
- **first** (*integer*) – Return locks starting from the specified offset. Default is `1`, the first lock.
- **number** (*integer*) – Return at most the specified number of locks. Default is `100`.
- **metadata** (*string[]*) – Filter out only the locks that has metadata according to the filter criteria.
  - *key* = *value* - Multiple query parameters can be specified.

  Note that = is part of the query parameter, and has to be encoded (`%3d`).
- **username** (*string*) – Comma-separated user names. Filter only locks created by the specified user(s).
- **range** (*string*) – Filter out locks whose expiry time is within the specified range. The range format is `[d..d]`, `(d..d)`, `[d..d)`, `(d..d]`, `(*..d]`, `[d..*)`, or `(*..*)`. `d` is a date and time in the ISO 8601 format.

**Produces**

- **application/xml, application/json** – DeletionLockListDocument

**Role** _deletion_lock_read

#### Retrieve a lock for an entity

**GET {lock-entity}/deletion-lock/** (*lock-id*)
Returns a specific lock.

**Produces**

- **application/xml, application/json** – XML/JSON, schema DeletionLockDocument

**Role** _deletion_lock_read

### 17.4.2 Managing Deletion Locks

#### Create a lock

**POST {lock-entity}/deletion-lock**
Creates new deletion lock based on the information in the DeletionLockDocument.

**Accepts**

- **application/xml, application/json** – DeletionLockDocument

**Produces**

- **application/xml, application/json** – DeletionLockDocument

**Role** _deletion_lock_write

#### Example

```
POST /deletion-lock
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DeletionLockDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <expiryTime>2021-09-12T10:40:14.546+02:00</expiryTime>
  <metadata>
    <field>
      <key>workflow</key>
      <value>production</value>
    </field>
  </metadata>
</DeletionLockDocument>
```

### Update a lock

**PUT /deletion-lock/** (*lock-id*)

Updates a lock based on the information in the DeletionLockDocument.

**Accepts**

- **application/xml, application/json** – DeletionLockDocument

**Produces**

- **application/xml, application/json** – DeletionLockDocument

**Role** _deletion_lock_write

### Update a lock

**PUT {lock-entity}/deletion-lock/** (*lock-id*)

Updates a lock based on the information in the DeletionLockDocument.

**Accepts**

- **application/xml, application/json** – DeletionLockDocument

**Produces**

- **application/xml, application/json** – DeletionLockDocument

**Role** _deletion_lock_write

### Delete a lock

**DELETE /deletion-lock/** (*lock-id*)
**DELETE {lock-entity}/deletion-lock/** (*lock-id*)

Delete a lock that was explicitly set on this entity.

**Role** _deletion_lock_write

## 17.5 Configuration

The configuration resource contains the system wide configuration that would typically be tuned by an administrator or set once when installing Vidispine and your application on a new system.

**See also:**

See *Configuration properties* for more information about the available configuration properties.

### 17.5.1 Configuration resources

**GET /configuration**
Returns the available configuration resource endpoints.

**Produces**

- **application/xml, application/json** – URIListDocument containing the names of the endpoints.
- **text/plain** – CRLF-delimited list of names

### 17.5.2 Indexing settings

#### Retrieve the indexing configuration

**GET /configuration/indexing**
Returns the current indexing configuration.

**Produces**

- **application/xml, application/json** – IndexingConfigurationDocument

**Role** _administrator

#### Update the indexing configuration

**PUT /configuration/indexing**
Updates the indexing configuration.

**Status Codes**

- **200 OK** – The configuration was updated successfully.

**Accepts**

- **application/xml, application/json** – IndexingConfigurationDocument

**Role** _administrator

### 17.5.3 Metrics settings

See *Monitoring* for examples.

#### Retrieve the metrics configuration

**GET /configuration/metrics**
Returns the current metrics configuration.

**Produces**

- **application/xml, application/json** – MetricsConfigurationDocument

**Role** _administrator

#### Update the metrics configuration

**PUT /configuration/metrics**
Updates the metrics configuration.

**Status Codes**

- **200 OK** – The configuration was updated successfully.

**Accepts**

- **application/xml, application/json** – MetricsConfigurationDocument

**Role** _administrator

### 17.5.4 Path alias configuration

See *Content paths* for information on paths and aliases.

#### Retrieve the path alias configuration

**GET /configuration/path-alias**
Returns the current path alias configuration.

**Produces**

- **application/xml, application/json** – PathAliasConfigurationDocument

**Role** _administrator

#### Example

```
GET /configuration/path-alias
```

```
<PathAliasConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <alias>v(name)=metadata.timespan[start=-INF][end=+INF].field[name=$name].value.value
↪</alias>
</PathAliasConfigurationDocument>
```

#### Update the path alias configuration

**PUT /configuration/path-alias**
Updates the path alias configuration.

**Status Codes**

- **200 OK** – The configuration was updated successfully.

**Accepts**

- **application/xml, application/json** – PathAliasConfigurationDocument

**Role** _administrator

#### Example

```
PUT /configuration/path-alias
Content-Type: application/xml

<PathAliasConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <alias>v(name)=metadata.timespan[start=-INF][end=+INF].field[name=$name].value.value
↪</alias>
  <alias>detail(tag)=shape[tag=$tag].containerComponent.format,shape[tag=$tag].
↪videoComponent.[codec,duration]</alias>
</PathAliasConfigurationDocument>
```

```
200 OK
```

### 17.5.5 Job pool configuration

#### Retrieve the job pool configuration

**GET /configuration/job-pool**
Returns the current job pool configuration.

**Produces**

- **application/xml, application/json** – JobPoolListDocument

**Role** _administrator

#### Example

```
GET /configuration/job-pool
```

```
<JobPoolListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <maxConcurrent>3</maxConcurrent>
</JobPoolListDocument>
```

#### Update the job pool configuration

**PUT /configuration/job-pool**
Updates the job pool configuration.

**Accepts**

- **application/xml, application/json** – JobPoolListDocument

**Role** _administrator

#### Example

```
PUT /configuration/job-pool
Content-Type: application/xml

<JobPoolListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <maxConcurrent>5</maxConcurrent>
  <pool>
    <priorityThreshold>HIGH</priorityThreshold>
    <size>2</size>
  </pool>
  <pool>
    <priorityThreshold>MEDIUM</priorityThreshold>
    <size>3</size>
  </pool>
</JobPoolListDocument>
```

#### Delete all job pools

**DELETE /configuration/job-pool**
Deletes all job pools.

Note that the max concurrent jobs setting will *not* be affected.

**Role** _administrator

### Example

```
GET /configuration/job-pool
```

```
<JobPoolListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <maxConcurrent>5</maxConcurrent>
  <pool>
    <priorityThreshold>HIGH</priorityThreshold>
    <size>2</size>
  </pool>
  <pool>
    <priorityThreshold>MEDIUM</priorityThreshold>
    <size>3</size>
  </pool>
</JobPoolListDocument>
```

```
DELETE /configuration/job-pool
```

```
GET /configuration/job-pool
```

```
<JobPoolListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <maxConcurrent>5</maxConcurrent>
</JobPoolListDocument>
```

### Delete a job pool

**DELETE /configuration/job-pool/** (*priority*)

Deletes the job pool with the given priority threshold.

**Role** _administrator

### Example

```
GET /configuration/job-pool
```

```
<JobPoolListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <maxConcurrent>5</maxConcurrent>
  <pool>
    <priorityThreshold>HIGH</priorityThreshold>
    <size>2</size>
  </pool>
  <pool>
    <priorityThreshold>MEDIUM</priorityThreshold>
    <size>3</size>
  </pool>
</JobPoolListDocument>
```

```
DELETE /configuration/job-pool/MEDIUM
```

```
GET /configuration/job-pool
```

```
<JobPoolListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <maxConcurrent>5</maxConcurrent>
  <pool>
```

```
    <priorityThreshold>HIGH</priorityThreshold>
    <size>2</size>
  </pool>
</JobPoolListDocument>
```

### 17.5.6 FTP pool configuration

#### Retrieve the FTP pool configuration

**GET /configuration/ftp-pool**
Returns the current FTP connection pool configuration.

**Produces**

- **application/xml, application/json** – FtpPoolConfigurationDocument

**Role** _administrator

#### Example

```
GET /configuration/ftp-pool
```

```
<FtpPoolConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <pool/>
</FtpPoolConfigurationDocument>
```

#### Update the FTP pool configuration

**PUT /configuration/ftp-pool**
Updates the FTP connection pool configuration.

**Accepts**

- **application/xml, application/json** – FtpPoolConfigurationDocument

**Role** _administrator

#### Example

```
PUT /configuration/ftp-pool
Content-Type: application/xml

<FtpPoolConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <pool>
    <minSize>0</minSize>
    <maxSize>-1</maxSize>
    <evictionInterval>30000</evictionInterval>
    <minIdleTime>60000</minIdleTime>
  </pool>
</FtpPoolConfigurationDocument>
```

#### Delete the FTP pool

**DELETE /configuration/ftp-pool**
Deletes the FTP connection pool.

**Role** _administrator

#### Example

```
GET /configuration/ftp-pool
```

```
<FtpPoolConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <pool/>
</FtpPoolConfigurationDocument>
```

```
DELETE /configuration/ftp-pool
```

```
GET /configuration/ftp-pool
```

```
<FtpPoolConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine"/>
```

### 17.5.7 Log report configuration

#### Retrieve the log report configuration

**GET /configuration/logreport**
Returns the current LogReport configuration.

**Produces**

- **application/xml, application/json** – LogReportConfigurationDocument

**Role** _administrator

#### Update the log report configuration

**PUT /configuration/logreport**
Updates the LogReport configuration.

**Status Codes**

- **200 OK** – The configuration was updated successfully.

**Accepts**

- **application/xml, application/json** – LogReportConfigurationDocument

**Role** _administrator

### 17.5.8 CORS configuration

New in version 4.15.

See *CORS configuration* for examples.

#### Retrieve the CORS configuration

**GET /configuration/cors**
Returns the current CORS configuration.

**Produces**

- **application/xml, application/json** – CORSConfigurationDocument

**Role** _administrator

#### Update the CORS configuration

**PUT /configuration/cors**
Updates the CORS configuration.

**Status Codes**

- **200 OK** – The configuration was updated successfully.

**Accepts**

- **application/xml, application/json** – CORSConfigurationDocument

**Role** _administrator

### 17.5.9 Database purging configuration

#### Retrieve the database purging configuration

**GET /configuration/purging**
Returns the current database purging configuration.

**Produces**

- **application/xml, application/json** – DatabasePurgingConfigurationDocument

**Role** _administrator

#### Update the database purging configuration

**PUT /configuration/purging**
Updates the database purging configuration. Note that if a category element is missing, e.g. `auditTrail`, that category is left unchanged. To remove a particular category, use an empty element, `<auditTrail/>`.

**Status Codes**

- **200 OK** – The configuration was updated successfully.

**Accepts**

- **application/xml, application/json** – DatabasePurgingConfigurationDocument

**Role** _administrator

#### Example

```
PUT /configuration/purging
Content-Type: application/xml

<DatabasePurgingConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine
↪">
  <auditTrail>
    <age>1440</age>
    <uri>ftp://user:password@myhost/logs/</uri>
    <compress>true</compress>
  </auditTrail>
</DatabasePurgingConfigurationDocument>
```

#### Remove the database purging configuration

**DELETE /configuration/purging**
Removes all database purging configuration.

**Status Codes**

- **200 OK** – The configuration was updated successfully.

**Role** _administrator

### 17.5.10 Default job priority configuration

New in version 5.2.1.

#### Retrieve the default job priority configuration

**GET /configuration/job-priority**
Returns the current default job priority configuration.

**Produces**

- **application/xml, application/json** – JobPriorityConfigurationDocument

**Role** _administrator

#### Update the default job priority configuration

**PUT /configuration/job-priority**
Updates the default job priority configuration.

**Status Codes**

- **200 OK** – The configuration was updated successfully.

**Accepts**

- **application/xml, application/json** – JobPriorityConfigurationDocument

**Role** _administrator

#### Example

```
PUT /configuration/job-priority
Content-Type: application/xml

<JobPriorityConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <job type="IMPORT">MEDIUM</job>
  <job type="EXPORT">HIGH</job>
</JobPriorityConfigurationDocument>
```

#### Remove the default job priority configuration

**DELETE /configuration/job-priority**
Removes all database purging configuration.

**Status Codes**

- **200 OK** – The configuration was updated successfully.

**Role** _administrator

### 17.5.11 OAuth2 configuration

New in version 4.17.

See *Configure OAuth2 using the API* for examples.

#### Retrieve the OAuth2 configuration

**GET /configuration/auth**
Returns the current OAuth2 configuration.

**Produces**

- **application/xml, application/json** – OAuth2ConfigurationDocument

**Role** _administrator

#### Update the OAuth2 configuration

**PUT /configuration/auth**
Updates the OAuth2 configuration.

**Status Codes**

- **200 OK** – The configuration was updated successfully.

**Accepts**

- **application/xml, application/json** – OAuth2ConfigurationDocument

**Produces**

- **application/xml, application/json** – OAuth2ConfigurationDocument

**Role** _administrator

#### Delete the OAuth2 configuration

**DELETE /configuration/auth**
Deletes and resets the current OAuth2 configuration.

**Role** _administrator

### 17.5.12 Bulky metadata storage configuration

New in version 5.3.

See *Bulky metadata storage* for examples.

#### Retrieve the bulky metdata storage configuration

**GET /configuration/bulkymetadata**
Returns the current bulky metadata configuration, together with some status information.

**Produces**

- **application/xml, application/json** – BulkyMetadataConfigurationDocument

**Role** _administrator

#### Update the bulky metdata storage configuration

**PUT /configuration/bulkymetadata**
Updates the bulky metadata configuration.

**Status Codes**

- **200 OK** – The configuration was updated successfully.

**Accepts**

- **application/xml, application/json** – BulkyMetadataConfigurationDocument

**Role** _administrator

### 17.5.13 Configuration properties

#### List all configuration properties

**GET /configuration/properties**
Returns a document containing all configuration properties set in the system.

**Produces**

- **application/xml, application/json** – ConfigurationPropertyListDocument

**Role** _administrator

#### Example

```
GET /configuration/properties
```

```
<ConfigurationPropertyListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <property lastChange="2014-06-03T15:18:49.608+02:00">
    <key>apiuri</key>
    <value>http://vs.example.com:8080/API</value>
  </property>
</ConfigurationPropertyListDocument>
```

#### Retrieve a configuration property

**GET /configuration/properties/** (*key*)
Returns a document or string containing all current setting for a configuration property.

**Status Codes**

- **200 OK** – The value is returned
- **404 Not found** – The configuration property is not set

**Produces**

- **application/xml, application/json** – ConfigurationPropertyDocument
- **text/plain** – String value

**Role** _administrator

### Example

```
GET /configuration/properties/apiuri
Accept: application/xml
```

```
<ConfigurationPropertyDocument xmlns="http://xml.vidispine.com/schema/vidispine"␣
↪lastChange="2014-06-03T15:18:49.608+02:00">
  <key>apiuri</key>
  <value>http://vs.example.com:8080/API</value>
</ConfigurationPropertyDocument>
```

```
GET /configuration/properties/apiuri
Accept: text/plain
```

```
http://vs.example.com:8080/API
```

## Create/update a configuration property

**PUT /configuration/properties**

Creates or updates a configuration property.

**Status Codes**

- **200 OK** – The configuration property was created/modified successfully.

**Accepts**

- **application/xml, application/json** – ConfigurationPropertyDocument

**Role** _administrator

### Example

```
PUT /configuration/properties
```

```
<ConfigurationPropertyDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <key>apiuri</key>
  <value>http://127.0.0.1:18080/API/</value>
</ConfigurationPropertyDocument>
```

## Create/update multiple configuration properties

New in version 4.17.

**POST /configuration/properties**

Creates or updates multiple configuration properties at once, using a ConfigurationPropertyListDocument.

**Status Codes**

- **200 OK** – The configuration properties were created/modified successfully.

**Accepts**

- **application/xml, application/json** – ConfigurationPropertyListDocument

**Role** _administrator

### Example

```
POST /configuration/properties
```

```
<ConfigurationPropertyListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
 <property>
  <key>apiuri</key>
  <value>http://vidispine.example.com:8080/API/</value>
 </property>
 <property>
  <key>noauthuri</key>
  <value>http://noauth.example.com:8080/</value>
 </property>
</ConfigurationPropertyListDocument>
```

### Create/update a configuration property

**PUT /configuration/properties/** (*key*)
Creates or updates a configuration property.

**Status Codes**

- **200 OK** – The configuration property was created/modified successfully.

**Accepts**

- **text/plain** – String value

**Role** _administrator

### Example

```
PUT /configuration/properties/apiuri
```

```
http://127.0.0.1:18080/API/
```

### Delete a configuration property

**DELETE /configuration/properties/** (*key*)
Removes a configuration property.

**Status Codes**

- **200 OK** – The configuration property was successfully deleted

**Role** _administrator

### Example

```
DELETE /configuration/properties/example_property
```

```
200 OK
```

## 17.6 Export locations

It is possible to pre-define named export locations. When starting an export job, the location name can be passed as a parameter, the files will then be exported to the URI associated with the export location.

### 17.6.1 Managing export locations

#### List all export locations

**GET /export-location**
List all defined export locations.

**Produces**

- **application/xml, application/json** – ExportLocationListDocument

**Role** _export

#### Update or create an export location

**PUT /export-location/** (*location-name*)
Create a new export location, or if there already is one with that name, update it.

**Accepts**

- **application/xml, application/json** – ExportLocationDocument

**Produces**

- **application/xml, application/json** – ExportLocationDocument

**Role** _export

#### Example

Creating a new export location:

```
PUT /export-location/External_FTP
Content-Type: application/xml

<ExportLocationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>ftp://user:password@10.2.23.25/export/</uri>
</ExportLocationDocument>
```

#### Retrieve an export location

**GET /export-location/** (*location-name*)
Return information about the export location with the specified name.

**Produces**

- **application/xml, application/json** – ExportLocationDocument

**Role** _export

#### Delete an export location

**DELETE /export-location/** (*location-name*)
Delete the export location with the specified name.

**Role** _export

### 17.6.2 Export location script

#### Retrieve the export location script

**GET /export-location/** (*location-name*) **/script**
Retrieves the script on an export location.

**Status Codes**

- **404 Not found** – If the location has no script.

**Produces**

- **text/plain** –

**Role** _export

#### Update the export location script

**PUT /export-location/** (*location-name*) **/script**
Updates the script of an existing export location.

**Accepts**

- **text/plain** –

**Produces**

- **text/plain** – The script that was set.

**Role** _export

## 17.7 External identifiers

### 17.7.1 Managing external id namespaces

Manage external identifier namespaces.

#### Retrieve all known namespaces

**GET /external-id**
Retrieves all known external id namespaces.

**Produces**

- **application/xml, application/json** – ExternalIdentifierNamespaceListDocument

**Role** _administrator

#### Example

```
GET /external-id
```

```
<ExternalIdentifierNamespaceListDocument xmlns="http://xml.vidispine.com/schema/
↪vidispine">
  <namespace>
    <name>uuid</name>
    <pattern>[A-Fa-f0-9]{8}\-[A-Fa-f0-9]{4}\-[A-Fa-f0-9]{4}\-[A-Fa-f0-9]{4}\-[A-Fa-f0-
↪9]{12}</pattern>
  </namespace>
</ExternalIdentifierNamespaceListDocument>
```

#### Retrieve a specific namespace

**GET /external-id/** (*namespace-id*)
Retrieves the namespace with the specified name.

**Produces**

- **application/xml, application/json** – ExternalIdentifierNamespaceDocument

**Role** _administrator

### Example

```
GET /external-id/uuid
```

```
<ExternalIdentifierNamespaceDocument xmlns="http://xml.vidispine.com/schema/vidispine
↪">
  <name>uuid</name>
  <pattern>[A-Fa-f0-9]{8}\-[A-Fa-f0-9]{4}\-[A-Fa-f0-9]{4}\-[A-Fa-f0-9]{4}\-[A-Fa-f0-9]
↪{12}</pattern>
</ExternalIdentifierNamespaceDocument>
```

## Update or create a namespace

**PUT /external-id/** (*namespace-id*)

Creates or modifies a namespace with the specified name.

**Accepts**

- **application/xml, application/json** – ExternalIdentifierNamespaceDocument

**Role** _administrator

### Example

```
PUT /external-id/uuid
Content-Type: application/xml

<ExternalIdentifierNamespaceDocument xmlns="http://xml.vidispine.com/schema/vidispine
↪">
    <pattern>[A-Fa-f0-9]{8}\-[A-Fa-f0-9]{4}\-[A-Fa-f0-9]{4}\-[A-Fa-f0-9]{4}\-[A-Fa-f0-
↪9]{12}</pattern>
</ExternalIdentifierNamespaceDocument>
```

```
200 OK
```

## Delete a namespace and all external ids in that namespace

**DELETE /external-id/** (*namespace-id*)

Deletes the specified namespace together with all *external ids that exist in that namespace*.

**Role** _administrator

### Example

```
DELETE /external-id/uuid
```

```
200 OK
```

### 17.7.2 Managing external ids

Manage external ids.

The current supported resources can be seen in the table below. These are referred to as `{external-id-resource}` in the definitions below.

| Type | Path |
|---|---|
| Item | `/item/{item-id}/external-id` |
| Collection | `/collection/{collection-id}/external-id` |
| Library | `/library/{library-id}/external-id` |
| Job | `/job/{job-id}/external-id` |
| Notification | `.../notification/{notification-id}/external-id` |
| Storage | `/storage/{storage-id}/external-id` |
| Metadata-field | `/metadata-field/{field-name}/external-id` |
| Field-group | `/metadata-field/field-group/{field-group-name}/external-id` |
| Quota rule | `/quota/{rule-id}/external-id` |

#### Retrieve all external ids for an entity

**GET {external-id-resource}**
Retrieves all external ids that are assigned to a particular entity.

**Produces**

- **application/xml, application/json** – ExternalIdentifierListDocument

**Role** _external_id_read

#### Example

```
GET /storage/VX-1/external-id
```

```
<ExternalIdentifierListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>
    <entityId>VX-1</entityId>
    <entityType>Storage</entityType>
    <namespace>uuid</namespace>
    <externalId>38eebf93-2ab7-463b-ba3a-b6217bb5bca9</externalId>
  </id>
</ExternalIdentifierListDocument>
```

```
GET /storage/38eebf93-2ab7-463b-ba3a-b6217bb5bca9/external-id
```

```
<ExternalIdentifierListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>
    <entityId>VX-1</entityId>
    <entityType>Storage</entityType>
    <namespace>uuid</namespace>
    <externalId>38eebf93-2ab7-463b-ba3a-b6217bb5bca9</externalId>
  </id>
</ExternalIdentifierListDocument>
```

#### Create a new external id

**PUT {external-id-resource}/**(*external-id*)
Creates a new external id for the specified entity.

**Role** _external_id_write

---

**Note:** External ids can only contain a maximum amount of 100 characters, regardless of the max size that is specified in the namespace pattern.

---

### Example

```
PUT /storage/VX-1/external-id/38eebf93-2ab7-463b-ba3a-b6217bb5bca9
```

```
200 OK
```

```
PUT /storage/VX-1/external-id/38eebf93-2ab7-463b-ba3a-b6217bb5bca9
```

```
400 An invalid parameter was entered
Context: external-id
Reason: That external id is already in use by VX-1.
```

### Clear all external ids for an entity

**DELETE {external-id-resource}**
Clears all external identifiers that are registered with an entity.

**Role** _external_id_write

### Example

```
DELETE /storage/VX-1/external-id
```

```
200 OK
```

```
GET /storage/38eebf93-2ab7-463b-ba3a-b6217bb5bca9/external-id
```

```
404 A resource could not be found
Type: external-id
ID: uuid_38eebf93-2ab7-463b-ba3a-b6217bb5bca9
```

### Remove an external id

**DELETE /external-id/id/**(*external-id*)
Removes the external identifier from all entities.

**Role** _external_id_write

**DELETE {external-id-resource}/**(*external-id*)
Removes the external identifier from a specific entity.

**Role** _external_id_write

### Example

```
DELETE /external-id/id/38eebf93-2ab7-463b-ba3a-b6217bb5bca9
```

```
200 OK
```

```
GET /storage/38eebf93-2ab7-463b-ba3a-b6217bb5bca9/external-id
```

```
404 A resource could not be found
Type: external-id
ID: uuid_38eebf93-2ab7-463b-ba3a-b6217bb5bca9
```

## 17.8 Groups and roles

Manage groups and roles.

### 17.8.1 Managing groups

#### List all groups/roles

**GET /group**
Returns list of all groups.

**Query Parameters**

- **first** (*integer*) – Start returning groups from specified number. Default is `1`, the beginning of the list.
- **number** (*integer*) – Return at most specified number of groups. Default is no limit.
- **role** (*boolean*) –
  - `true` - Return only roles.
  - `false` - Return only regular groups.

  Default is to return all.

  New in version 4.16.

**Produces**

- **application/xml, application/json** – GroupListDocument
- **text/plain** – CRLF-delimited list of group URIs

**Role** _group_read

#### Retrieve a group/role

**GET /group/** (*group-name*)
Returns information about the specified group.

**Produces**

- **application/xml, application/json** – GroupDocument
- **text/plain** – The URI to the group.

**Role** _group_read

### Retrieve role status

**GET /group/**(*group-name*)**/role**
Returns the role status of the specified group.

**Produces**

- **text/plain** – 1 if group is a role, 0 if group is a regular group

**Role** _group_read

### Create a group

**PUT /group/**(*group-name*)
Creates a new group with the specified name.

**Status Codes**

- **200 OK** – Group created.
- **409 Conflict** – A group with that name already exists.

**Role** _group_write

### Update or create a group

**PUT /group/**(*group-name*)
Creates or updates the group with the specified name. Also any specified parent and child associations, users, metadata and description will be added.

**Query Parameters**

- **clear** (*boolean*) –
  - `true` - Remove any existing groups and users not in the given document.
  - `false` (default) - Existing groups and users not in the request document will be kept as is.

**Status Codes**

- **200 OK** – Group created.

**Accepts**

- **application/xml, application/json** – GroupDocument

**Role** _group_write

### Delete a group

**DELETE /group/**(*group-name*)
Deletes the group with the specified name.

**Role** _group_write

### Delete multiple groups

**DELETE /group**
Deletes the groups with the specified names.

**Query Parameters**

- **name** (*string*) – Required. Comma-separated list of group names.

**Role** _group_write

### Example

```
DELETE /group?name=teachers,students,guests
```

### Search for groups

**PUT /group**
Simple search of fields `groupname`, `description` and metadata.

**Query Parameters**

- **first** (*integer*) – Start returning groups from specified number. Default is `1`, the beginning of the list.
- **number** (*integer*) – Return at most specified number of groups. Default is `10`.
- **role** (*boolean*) –
  - `true` - Return only roles.
  - `false` - Return only regular groups.

  Default is to return all.

**Accepts**

- **application/xml, application/json** – GroupSearchDocument

**Produces**

- **application/xml, application/json** – GroupListDocument
- **text/plain** – CRLF-delimited list of group names

### Example

```
PUT /group
Content-Type: application/xml

<GroupSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <field>
        <name>groupname</name>
        <value>vidi</value>
    </field>
    <field>
        <name>key</name>
        <value>value</value>
    </field>
</GroupSearchDocument>
```

Note that keywords `groupname` and `description` are reserved to do search on `groupname` and `description` fields

The boolean operators AND and OR are supported:

```
<GroupSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <field>
        <name>groupname</name>
        <value>vidi</value>
    </field>
    <field>
        <name>description</name>
```

```
        <value>vidispine</value>
    </field>
    <operator operation="OR">
        <field>
          <name>key1</name>
          <value>value1</value>
        </field>
        <field>
          <name>key2</name>
          <value>value2</value>
        </field>
    </operator>
</GroupSearchDocument>
```

### 17.8.2 Group information

#### Retrieve the group description

**GET /group/** (*group-name*) **/description**
Returns the descriptive text about the specified group.

**Produces**

- **text/plain** – Group description

**Role** _group_read

#### Update the description of a group

**PUT /group/** (*group-name*) **/description**
Changes the description of a group.

**Accepts**

- **text/plain** – The new description.

**Role** _group_write

### 17.8.3 Group-to-group relations

#### List all parent groups to a group

**GET /group/** (*group-name*) **/parents**
Returns groups that the specified group belongs to.

**Query Parameters**

- **traverse** (*boolean*) –
  - `true` - Return all ancestors.
  - `false` (default) - Return only direct parents.

**Produces**

- **application/xml, application/json** – GroupListDocument
- **text/plain** – CRLF-delimeted list of URIs to the groups

**Role** _group_read

#### List all child groups to a group

**GET /group/**(*group-name*)**/children**
Returns groups that belongs to the specified group.

**Query Parameters**

- **traverse** (*boolean*) –
  - `true` - Return all descendants.
  - `false` (default) - Return only direct children.

**Produces**

- **application/xml, application/json** – GroupListDocument
- **text/plain** – CCRLF-delimeted list of URIs to the groups

**Role** _group_read

#### Add a group to another group

**PUT /group/**(*group-name*)**/group/**
*child-groupname* Creates parent-child relation between the two specified groups.

**Role** _group_write

#### Remove a group from another group

**DELETE /group/**(*group-name*)**/group/**
*child-groupname* Removes the parent-child relation between the two specified groups.

**Role** _group_write

### 17.8.4 Group-to-user relations

#### List all users in a group

**GET /group/**(*group-name*)**/users**
Returns all users belonging to the group/role, directly or indirectly.

**Query Parameters**

- **traverse** (*boolean*) –
  - `true` - Return all users, including users in child groups.
  - `false` (default) - Return only direct users.

**Produces**

- **application/xml, application/json** – UserListDocument
- **text/plain** – CRLF-delimeted list of *Tabbed tuples* of user name, user real name

**Role** _group_read

#### Add a user to a group

**PUT /group/**(*group-name*)**/user/**
*username* Adds the specified user to the specified group.

**Role** _group_write

#### Remove a user from a group

**DELETE /group/**(*group-name*)**/user/**
*username* Removes the specified user from the specified group.

**Role** _group_write

## 17.9 Imports

Trigger imports of files, sidecar files and IMF packages.

### 17.9.1 Importing an item

An item can be imported in two ways, either through supplying a URI or sending the data in the request body.

There is also a third automatic way, using *Automatic import*.

#### Import using a URI

**POST /import**
Starts a *job* that imports the file, located at the given URI, and creates an item. Note that thumbnails and poster frames are only generated if a transcode takes place.

**Query Parameters**

- **uri** (*string*) – Required. A URI to the file that will be imported. Make sure to *percent encode* the URI.
- **URL** (*string*) – A URL to the file that will be imported. (Deprecated since 4.2.)
- **tag** (*string[]*) – A list of *shape tags* to use for transcoding.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **no-transcode** (*boolean*) –
  - `true` - Will disable transcoding even if the `tags` parameter is set. Rather, the specified tag will be used to determine cropping, scaling etc. of thumbnails.
  - `false` (default) - Normal transcode.
- **no-mediacheck** (*boolean*) – Only for for import of mpeg-dash mpd, skip media check to speed up import.
  - `true` - Will disable media check.
  - `false` (default) - Normal media check.
- **createThumbnails** (*boolean*) –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **overrideFastStart** (*boolean*) –
  - `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.

  - `false` - Do not use the transcoder's estimate of the duration.
- **requireFastStart** (*boolean*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** (*string*) – Estimated duration of the clip in seconds.
- **storageId** (*string*) – Identifier of storage where essence file is to be stored.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **growing** (*boolean*) –
  - `true` - Specifies that the input file is still written to, so enables growing file support.
  - `false` (default) - No growing file handling of import file.
- **xmpfile** (*string*) – URI to a sidecar XMP metadata file.
- **sidecar** (*string*) – URIs or file ids of any sidecar files to import to the item.
- **settings** (*string*) – Pre-configured *import settings*.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **importTag** (*string[]*) – A list of shape tags that the created shape will be associated with. Default is `original`.

**Accepts**

- **application/xml, application/json** – MetadataDocument, initial metadata that is given to the imported item.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

### Example

```
POST /import?uri=http://example.com/video.avi HTTP/1.1
Accept: application/xml
Content-type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
      <timespan end="+INF" start="-INF">
         <field>
            <name>title</name>
            <value>This is an imported item!</value>
         </field>
      </timespan>
</MetadataDocument>
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <jobId>VX-80</jobId>
    <status>READY</status>
    <type>PLACEHOLDER_IMPORT</type>
</JobDocument>
```

### Import using the request body

**POST /import/raw**

Starts a job that reads the raw data from the request body, generates a file, and imports the file.

**Query Parameters**

- **tag** (*string[]*) – A list of *shape tags* to use for transcoding.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **no-transcode** (*boolean*) –
  - `true` - Will disable transcoding even if the `tags` parameter is set. Rather, the specified tag will be used to determine cropping, scaling etc. of thumbnails.
  - `false` (default) - Normal transcode.
- **createThumbnails** (*boolean*) –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **transferPriority** (*integer*) – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers. A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **transferId** (*string*) – An id to assign the transfer to be able to refer to it.
- **storageId** (*string*) – Identifier of storage where essence file is to be stored.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **overrideFastStart** (*boolean*) –
  - `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
  - `false` - Do not use the transcoder's estimate of the duration.
- **requireFastStart** (*boolean*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** (*string*) – Estimated duration of the clip in seconds.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.

- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **filename** (*string*) – The filename to be stored as original filename
- **settings** (*string*) – Pre-configured *import settings*.
- **ids** (*string*) – Comma-separated list of external ids to assign to the item.
- **importTag** (*string[]*) – A list of shape tags that the created shape will be associated with. Default is `original`.

**Status Codes**

- **400** – If the amount of data received does not match the given Content-Length header.

**Request Headers**

- **index** – Offset (in bytes) of the full file for where the first byte of this transfer is located.
- **size** – The total size of the full file.

**Accepts**

- **application/octet-stream** – The raw data.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job, or no content if the transfer is not finished.

**Role** _import

### Semantics

There are two modes of operation for this type of import. The most simple is to transfer the entire file and then the header parameters can be ignored. The other is to transfer the file over multiple requests, then the header parameters are required. If the latter mode is used, then the job will not start until the entire file is transferred.

Note that thumbnails and poster frames are only generated if a transcode takes place.

---

**Tip:** *Managing transfers*

Transfers can be managed, see *Transfers*.

---

### Example: transferring the entire file

```
POST /import/raw HTTP/1.1
Content-Type: application/octet-stream

<the entire file data>
```

### Example: transferring a file using multiple requests

Assume a file that is 1000 bytes. This file can be sent using three requests, where one request sends data [800, 1000], another sends data [0, 300] and the last request sends data [300, 800].

```
POST /import/raw?transferId=mytransfer HTTP/1.1
Content-Type: application/octet-stream
size: 1000
index: 800

<200 bytes of file data, starting at byte 800>
```

```
POST /import/raw?transferId=mytransfer HTTP/1.1
Content-Type: application/octet-stream
size: 1000
index: 0

<300 bytes of file data, starting at byte 0>
```

```
POST /import/raw?transferId=mytransfer HTTP/1.1
Content-Type: application/octet-stream
size: 1000
index: 300

<500 bytes of file data, starting at byte 300>
```

The last request that finishes will start the job and receive the corresponding job document.

### Import using a passkey

**POST /import/raw-passkey**

Create a job and generates a passkey that can later be used to import an item without being authenticated.

**Query Parameters**

- **tag** (*string[]*) – A list of *shape tags* to use for transcoding.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **no-transcode** (*boolean*) –
  - `true` - Will disable transcoding even if the `tags` parameter is set. Rather, the specified tag will be used to determine cropping, scaling etc. of thumbnails.
  - `false` (default) - Normal transcode.
- **createThumbnails** (*boolean*) –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **transferPriority** (*integer*) – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers. A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **transferId** (*string*) – Required. An id to assign the transfer to be able to refer to it.
- **storageId** (*string*) – Identifier of storage where essence file is to be stored.
- **overrideFastStart** (*boolean*) –

  - `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
  - `false` - Do not use the transcoder's estimate of the duration.
- **requireFastStart** (*boolean*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** (*string*) – Estimated duration of the clip in seconds.
- **filename** (*string*) – The filename to be stored as original filename
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **settings** (*string*) – Pre-configured *import settings*.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **ids** (*string*) – Comma-separated list of external ids to assign to the item.
- **importTag** (*string[]*) – A list of shape tags that the created shape will be associated with. Default is `original`.

**Status Codes**

- **400** – If the amount of data received does not match the given Content-Length header.

**Accepts**

- **application/xml, application/json** – MetadataDocument, initial metadata that is given to the imported item.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

### Example

```
POST /import/raw-passkey?transferId=mytransfer HTTP/1.1
Accept: application/xml
Content-type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
      <timespan end="+INF" start="-INF">
         <field>
            <name>title</name>
            <value>This is an imported item!</value>
         </field>
      </timespan>
</MetadataDocument>
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <jobId>VX-102</jobId>
    <status>WAITING</status>
    <type>RAW_IMPORT</type>
    <data>
        <key>passkey</key>
        <value>91df2b2fe74957cc7331d59a59a88cdc14df460dbb4d62c20287399b30092134</
↪value>
    </data>
</JobDocument>
```

### Importing without authentication

**Note:** Note that this request uses `http://server:port/APInoauth/...` instead of the usual `http://server:port/API/...`

**`POST /APInoauth/import/raw`**

Imports the item and starts the job.

**Query Parameters**

- **`tag`** (*`string[]`*) – A list of *shape tags* to use for transcoding.
- **`original`** (*`string`*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **`createThumbnails`** (*`boolean`*) –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **`thumbnailService`** (*`string`*) – Identifies which thumbnail resource that should be used.
- **`createPosters`** (*`string`*) – A list of *time codes* to use for creating posters.
- **`transferPriority`** (*`integer`*) – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers. A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **`transferId`** (*`string`*) – An id to assign the transfer to be able to refer to it.
- **`storageId`** (*`string`*) – Identifier of storage where essence file is to be stored.
- **`overrideFastStart`** (*`boolean`*) –
  - `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
  - `false` - Do not use the transcoder's estimate of the duration.
- **`requireFastStart`** (*`boolean`*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **`fastStartLength`** (*`string`*) – Estimated duration of the clip in seconds.
- **`filename`** (*`string`*) – The filename to be stored as original filename

- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **settings** (*string*) – Pre-configured *import settings*.

**Status Codes**

- **400** – If the amount of data received does not match the given Content-Length header.

**Accepts**

- **application/octet-stream, multipart/form-data** – The raw essence data. If sent as multipart form data, the file should then be sent in the `file` parameter.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

### Example

```
POST /import/raw?transferId=mytransfer&
 ↪passkey=91df2b2fe74957cc7331d59a59a88cdc14df460dbb4d62c20287399b30092134
Accept: application/xml
Content-type: application/octet-stream

<file data>
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <jobId>VX-102</jobId>
    <user>admin</user>
    <started>2010-08-11T09:57:29.575+02:00</started>
    <status>READY</status>
    <type>RAW_IMPORT</type>
    <priority>MEDIUM</priority>
</JobDocument>
```

### Import an IMF package using a URI

**POST /import/imp**

Starts a job that reads the asset map of an IMP (IMF package), and then imports the IMP. Essence files whose UUID is already managed by Vidispine Server are not copied. For more information about jobs, see *Jobs*. Note that thumbnails and poster frames are only generated if a transcode takes place.

Changed in version 5.3: IMF packages will now be validated using Photon and results saved as metadata on the item. Can be disabled with **jobMetadata** parameter.

**Query Parameters**

- **uri** (*string*) – Required. A URI to the asset map that will be imported. Tracks referenced by the asset map should be located in the same folder.
- **tag** (*string[]*) – A list of *shape tags* to use for transcoding.

- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **no-transcode** (*boolean*) –
  - `true` - Will disable transcoding even if the `tags` parameter is set. Rather, the specified tag will be used to determine cropping, scaling etc. of thumbnails.
  - `false` (default) - Normal transcode.
- **createThumbnails** (*boolean*) –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **overrideFastStart** (*boolean*) –
  - `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
  - `false` - Do not use the transcoder's estimate of the duration.
- **requireFastStart** (*boolean*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** (*string*) – Estimated duration of the clip in seconds.
- **storageId** (*string*) – Identifier of storage where essence file is to be stored.
- **growing** (*boolean*) –
  - `true` - Specifies that the input file is still written to, so enables growing file support.
  - `false` (default) - No growing file handling of import file.
- **xmpfile** (*string*) – URI to a sidecar XMP metadata file.
- **sidecar** (*string*) – URIs or file ids of any sidecar files to import to the item.
- **settings** (*string*) – Pre-configured *import settings*.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **importTag** (*string[]*) – A list of shape tags that the created shape will be associated with. Default is `original`.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **noCPLreimport** (*boolean*) – If `true`, do not allow reimport of CPL with same id. Default is `false`

**Accepts**

- **application/xml, application/json** – MetadataDocument, initial metadata that is given to the imported item.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

The **importtag** query parameter either be a list of shapes, or on the format `uuid=tag`. The IMF import job accepts certain special **jobMetadata** parameters:

**imfDisallowImportOfCPLWithoutShapeTag** If set to true, do not accept imports of CPLs which do not match any import tags (see `importtag` above).

**imfDisallowImportOfIMPWithMissingMedia** If set to true, do not accept imports of CPLs which contains references to media not in the CPL and not already in Vidispine.

**skipImpValidation** If set to true, do not perform Photon IMF package validation.

New in version 5.3.

#### Find destination storage

New in version 4.15.

**GET /import/storage**
Find where imported content would be stored according to current storage rules.

**Query Parameters**

- **item** (*string*) – Item id
- **tag** (*string*) – Shape tag
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – StorageDocument

**Role** _import

### 17.9.2 Placeholder imports

A placeholder import is an import where the item and a shape are created before any file is transferred. Once all the specified files have been transferred, an import job will start.

#### Create a placeholder item

**POST /import/placeholder**
Creates an empty item and a shape with components matching the given parameters.

**Query Parameters**

- **container** (*integer*) – The number of files that contain container components.
- **audio** (*integer*) – The number of files that contain audio components.
- **video** (*integer*) – The number of files that contain video components.
- **binary** (*integer*) – The number of files that contain binary components.
- **type** (*string*) –

- `image-sequence`- Image sequence.
- `dpx` - DPX sequence.

Deprecated since version 4.6: Import image sequences using URIs with file fragments instead.

- **`frameDuration`** (*`string`*) – *duration* for each image in the sequence.
- **`settings`** (*`string`*) – Pre-configured *import settings*.
- **`importTag`** (*`string[]`*) – A list of shape tags that the created shape will be associated with. Default is `original`.
- **`externalId`** (*`string`*) – An external identifier to assign to the item.

**Accepts**

- **`application/xml, application/json`** – MetadataDocument, initial metadata that is given to the imported item.

**Produces**

- **`text/plain`** – The id of the item
- **`application/xml, application/json`** – ItemDocument

**Role** _import

### Import to a placeholder item

**`POST /import/placeholder/`** (*item-id*) **`/`**

*component-type* Imports the file and extracts component data based on what type is specified (container, audio, video, binary). No transcoding will take place until all files have been imported.

**Query Parameters**

- **`uri`** (*`string`*) – A URI to the file that will be imported. Make sure to *percent encode* the URI. Must be specified unless `fileId` is specified.
- **`fileId`** (*`string`*) – The id of the file that contains the essence. Must be specified unless `uri` is specified.
- **`allowReimport`** (*`boolean`*) –
  - `true` - Import the file to this shape even if the file is already importing or is already part of another item.
  - `false` (default) Reject the request if the file with the given id has already been imported or is currently importing.
- **`tag`** (*`string[]`*) – A list of *shape tags* to use for transcoding.
- **`original`** (*`string`*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **`no-transcode`** (*`boolean`*) –
  - `true` - Will disable transcoding even if the `tags` parameter is set. Rather, the specified tag will be used to determine cropping, scaling etc. of thumbnails.
  - `false` (default) - Normal transcode.
- **`createThumbnails`** (*`boolean`*) –
  - `true` (default) - Generate thumbnails as per defined by shape tag.

  - `false` - Disable thumbnail generation.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **storageId** (*string*) – Identifier of storage where essence file is to be stored.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **overrideFastStart** (*boolean*) –
  - `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
  - `false` - Do not use the transcoder's estimate of the duration.
- **requireFastStart** (*boolean*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** (*string*) – Estimated duration of the clip in seconds.
- **growing** (*boolean*) –
  - `true` - Specifies that the input file is still written to, so enables growing file support.
  - `false` (default) - No growing file handling of import file.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **settings** (*string*) – Pre-configured *import settings*.
- **index** (*integer*) – The component index (track) of new component.
- **shapeId** (*string*) – Shape id for which shape to receive the content.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

### Import to a placeholder item using the request body

**POST /import/placeholder/** (*item-id*) **/** *component-type* **/raw** Imports the file and extracts component data based on what type is specified (container, audio, video, binary). No transcoding will take place until all files have been imported.

**Query Parameters**

- **tag** (*string[]*) – A list of *shape tags* to use for transcoding.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **no-transcode** (*boolean*) –

  - `true` - Will disable transcoding even if the `tags` parameter is set. Rather, the specified tag will be used to determine cropping, scaling etc. of thumbnails.
  - `false` (default) - Normal transcode.
- **`createThumbnails`** (*`boolean`*) –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **`thumbnailService`** (*`string`*) – Identifies which thumbnail resource that should be used.
- **`createPosters`** (*`string`*) – A list of *time codes* to use for creating posters.
- **`transferPriority`** (*`integer`*) – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers. A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **`transferId`** (*`string`*) – An id to assign the transfer to be able to refer to it.
- **`storageId`** (*`string`*) – Identifier of storage where essence file is to be stored.
- **`resourceId`** (*`string`*) – The transcoder resource to use to execute the transcode.
- **`overrideFastStart`** (*`boolean`*) –
  - `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
  - `false` - Do not use the transcoder's estimate of the duration.
- **`requireFastStart`** (*`boolean`*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **`fastStartLength`** (*`string`*) – Estimated duration of the clip in seconds.
- **`filename`** (*`string`*) – The filename to be stored as original filename
- **`notification`** (*`string`*) – The *placeholder job notification* to use for this job.
- **`notificationData`** (*`string`*) – Any additional data to include for *notifications* on this job.
- **`priority`** (*`string`*) – The priority to assign to the job. Default is `MEDIUM` .
- **`jobmetadata`** (*`string[]`*) – Additional *information* for the job task.
- **`index`** (*`integer`*) – Index (order) of the component.
- **`shapeId`** (*`string`*) – Shape id for which shape to receive the content.

**Status Codes**

- **`400`** – If the amount of data received does not match the given Content-Length header.

**Request Headers**

- **`index`** – Offset (in bytes) of the full file for where the first byte of this transfer is located.
- **`size`** – The total size of the full file.

**Accepts**

- **`application/octet-stream`** – The raw data.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job, or no content if the transfer is not finished.

**Role** _import

### Example

Creating a placeholder item that consists of one file.

```
POST /import/placeholder?container=1
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan end="+INF" start="-INF">
      <field>
      <name>title</name>
         <value>My placeholder import!</value>
      </field>
   </timespan>
</MetadataDocument>
```

```
VX-1134
```

```
POST /import/placeholder/VX-1134/container?tag=lowres&uri=http://example.com/video.avi
Content-Type: application/xml

<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <jobId>VX-1299</jobId>
   <user>admin</user>
   <started>2010-05-07T16:12:10.023+02:00</started>
   <status>READY</status>
   <type>PLACEHOLDER_IMPORT</type>
   <priority>MEDIUM</priority>
</JobDocument>
```

### Import to a placeholder item in bulk

**POST /import/placeholder/** (*item-id*)

Imports the files and extracts component data based on what type is specified (container, audio, video, binary). No transcoding will take place until all files have been imported.

**Query Parameters**

- **tag** (*string[]*) – A list of *shape tags* to use for transcoding.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **no-transcode** (*boolean*) –
  - `true` - Will disable transcoding even if the `tags` parameter is set. Rather, the specified tag will be used to determine cropping, scaling etc. of thumbnails.
  - `false` (default) - Normal transcode.
- **overrideFastStart** (*boolean*) –

  - `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
  - `false` - Do not use the transcoder's estimate of the duration.
- **requireFastStart** (*boolean*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** (*string*) – Estimated duration of the clip in seconds.
- **storageId** (*string*) – Identifier of storage where essence file is to be stored.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **growing** (*boolean*) –
  - `true` - Specifies that the input file is still written to, so enables growing file support.
  - `false` (default) - No growing file handling of import file.
- **settings** (*string*) – Pre-configured *import settings*.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **importTag** (*string[]*) – A list of shape tags that the created shape will be associated with. Default is `original`.
- **index** (*integer*) – The component index (track) of new component.
- **shapeId** (*string*) – Shape id for which shape to receive the content.

**Accepts**

- **application/xml, application/json** – A PlaceholderImportRequestDocument describing the files to import.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

### Create passkey for placeholder item

**POST /import/placeholder/** (*item-id*) **/raw-passkey**

Creates a new passkey for a specific placeholder item, that can be used to perform imports to this item without requiring authentication.

**Query Parameters**

- **shapeId** (*string*) – Shape id for which shape to receive the content.

**Produces**

- **application/xml, application/json** – URIListDocument

#### Example

```
POST /import/placeholder/VX-25182/raw-passkey
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <uri>a8c87f5fa49344ba0bc3575616e047da426991addd6b23d7f7c7b05a21a4a083</uri>
</URIListDocument>
```

#### Adopt stand-alone files

**POST /import/placeholder/** (*item-id*) **/**
*component-type***/adopt/***file-id* Adopt the file as a component in a placeholder item. The value of component-type is one of: `container`,`audio`,`video`,`binary`

**Query Parameters**

- **index** (*integer*) – Index (order) of the component.
- **shapeId** (*string*) – Shape id for which shape to receive the content.

**Role** _import

### 17.9.3 Importing sidecar files

Sidecar files can be imported and saved as metadata of an item using a sidecar import job. The supported sidecar file formats are:

- EVS (.evs)
- SCC (.scc)
- SRT (.srt)
- STL (.stl) - EBU STL format (EBU Tech 3264).
- XMP (.xmp)
- Vidispine metadata (.xml)

Note that this may be a lossy operation, that is, that only parts of the data from the sidecar file is saved in the item metadata, and that it may not be possible to recreate the original sidecar file from the item metadata.

#### Import a sidecar file

**POST /import/sidecar/** (*item-id*)
Starts a job that imports the sidecar file, located at the given URL, to the specified item.

**Query Parameters**

- **sidecar** (*string*) – Either the id of the sidecar file or a URL for locating it.
- **startTimeCode** (*string*) – The expected start time code of the content. Can be used when importing SCC subtitles to a placeholder item to adjust the absolute time code of the SCC to relative (zero-based) timecodes. This require the configuration property *useAbsoluteSccTimeCode* to be set to false (default).
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .

- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

**POST /import/sidecar/** (*item-id*) **/raw**

Starts a job that imports the sidecar file as HTTP request body. The sidecar file will be saved in one of the Vidispine storages.

**Query Parameters**

- **storageId** (*string*) – The id of the storage that the sidecar file will be saved in.
- **fileExtension** (*string*) – The extension of the file that this sidecar data is from. Used to identify the sidecar media type. Example: `srt`.
- **startTimeCode** (*string*) – The expected start time code of the content. Can be used when importing SCC subtitles to a placeholder item to adjust the absolute time code of the SCC to relative (zero-based) timecodes. This require the configuration property *useAbsoluteSccTimeCode* to be set to false (default).
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Status Codes**

- **400** – If the amount of data received does not match the given Content-Length header.

**Accepts**

- **application/octet-stream** – The raw data.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

## 17.10 Import settings

Import settings profiles can be used to add access controls to items that are being imported.

### 17.10.1 Managing import settings

#### Create an import profile

**POST /import/settings**

Creates a new settings profile with the given settings.

**Accepts**

- **application/xml, application/json** – An ImportSettingsDocument containing the settings profile.

**Produces**

- **application/xml, application/json** – An ImportSettingsDocument containing the the settings profile together with its id.

**Role** _import

### Example

```
POST /import/settings
Content-Type: application/xml

<ImportSettingsDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <access>
      <permission>READ</permission>
      <user>myuser</user>
   </access>
</ImportSettingsDocument>
```

```
<ImportSettingsDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <id>VX-4</id>
   <access>
      <permission>READ</permission>
      <user>myuser</user>
   </access>
</ImportSettingsDocument>
```

### List all import profiles

**GET /import/settings**
Retrieves a list of all profiles.

**Produces**

- **application/xml, application/json** – A URIListDocument containing the ids of all profiles.
- **text/plain** – CRLF-delimited list of ids

**Role** _import

### Example

```
GET /import/settings
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <uri>VX-1</uri>
   <uri>VX-2</uri>
   <uri>VX-3</uri>
   <uri>VX-4</uri>
</URIListDocument>
```

### Retrieve an import profile

**GET /import/settings/**(*settings-id*)
Retrieves the settings specified by a certain profile.

**Produces**

- **application/xml, application/json** – An ImportSettingsDocument containing the settings of the profile.

**Role** _import

### Example

```
GET /import/settings/VX-4
```

```
<ImportSettingsDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <id>VX-4</id>
   <access>
      <permission>READ</permission>
      <user>myuser</user>
   </access>
</ImportSettingsDocument>
```

### Update an import profile

**PUT /import/settings/** (*settings-id*)
Changes the settings of the specified profile.

**Accepts**

- **application/xml, application/json** – An ImportSettingsDocument with the new settings.

**Role** _import

### Example

```
PUT /import/settings/VX-4
Content-Type: application/xml

<ImportSettingsDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <access>
      <permission>WRITE</permission>
      <user>myuser</user>
   </access>
</ImportSettingsDocument>
```

```
200 OK
```

### Delete an import profile

**DELETE /import/settings/** (*settings-id*)
Deletes the profile with specified id.

**Role** _import

### Example

```
DELETE /import/settings/VX-4
```

```
200 OK
```

# 17.11 Items

## 17.11.1 Exports

An item export is the process of copying a file from storage to a location accessible by the system.

### Create export jobs

### Start an export job for a single item

**POST /item/** (*item-id*) **/export**

Creates a new export job that will copy a file to a remote location.

A shape tag can be specified to decide which shape that will be exported.

If the URI ends with a "/" the URI is assumed to describe a folder and the file will retain its existing filename. Otherwise it is assumed that the URI describes a file and that filename will be used.

**Query Parameters**

- **uri** (*string*) – A URI to the destination of the file.
- **locationName** (*string*) – The name of an *export location*.
- **metadata** (*boolean*) –
  - `true` - Metadata will also be exported to side-car XML file.
  - `false` (default) - No metadata is exported.
- **projection** (*string*) – Defines the projection to use when exporting the metadata.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **useOriginalFilename** (*boolean*) – If set to `true`, the file(s) will be exported with their original filename if available. Default is `false`.
- **useOriginalComponentFilename** (*boolean*) – If set to `true`, the file(s) will be exported with their original component filename if available.
- **tag** (*string*) – Finds a shape with the specified tag and uses that for export. If not specified, the system will attempt to use the original shape.
- **start** (*string*) – Defines a start *time code* for the media.
- **end** (*string*) – Defines an end *time code* for the media.
- **template** (*string*) – export template to use.
- **allowMissing** (*boolean*) –
  - `true` (default) - Job will be started and the missing files will be ignored.
  - `false` - Job will fail if there are missing files and the files could not be generated by transcoding. A shape tag should be specified.
- **track** (*string*) – Comma-separated list of item track ids. Can include wildcards, e.g. `A*`. Can also contain component ids. Default is `*`, all tracks/components.

- **version** (*string*) –
  - *essence-version-id* - Return shapes for a specified version.
  - `all` - Return shapes for all versions.
  - `latest` (default) - Return shapes for the latest version.
  - `latest-per-shapetag` - Return shapes with the highest essence version number per shape tag.
- **exportDashMpd** (*boolean*) – If set to `true` the dash-mpd manifest is exported together with the selected representations specified in tag. If no tags is specified all representations will be exported. The representation files is exported to the original filename. (New in 5.3.)

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _export

---

**Note:** *FTP active mode*

For FTP exports, active mode can be forced by adding `?passive=false` to the FTP URL. To set the client side ports used in active mode, set the configuration property `ftpActiveModePortRange`, the value should be a range, e.g. `42100-42200`. To set the client IP used in active mode, set the configuration property `ftpActiveModeIp`.

---

**Note:** *XMP rewrite*

By using the `jobmetadata` query parameter with `rewriteXMP=false` (remember to URL encode the =), any XMP metadata in the source file will *not* be updated with the XMP metadata of the item.

---

### Example

Create a new export job that transfers the file of a shape with the tag `flv`.

```
POST /item/VX-250/export?tag=flv&uri=file:/home/user/video/myvideo.flv
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <jobId>VX-1293</jobId>
   <user>admin</user>
   <started>2010-05-07T14:05:51.826+02:00</started>
   <status>READY</status>
   <type>EXPORT</type>
   <priority>MEDIUM</priority>
</JobDocument>
```

### Start an export job for a single item as an IMF package

**POST /item/** (*item-id*) **/export/imp**

Creates a new export job that will create an IMF package as a remote location. URI must end with an "/" to denote a folder.

**Query Parameters**

- **uri** (*string*) – A URI to the destination of the file.
- **locationName** (*string*) – The name of an *export location*.

- **metadata** (*boolean*) –
  - `true` - Metadata will also be exported to side-car XML file.
  - `false` (default) - No metadata is exported.
- **projection** (*string*) – Defines the projection to use when exporting the metadata.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **useOriginalFilename** (*boolean*) – If set to `true`, the file(s) will be exported with their original filename if available. Default is `false`.
- **useOriginalComponentFilename** (*boolean*) – If set to `true`, the file(s) will be exported with their original component filename if available.
- **tag** (*string*) – Finds a shape with the specified tag and uses that for export. If not specified, the system will attempt to use the original shape.
- **start** (*string*) – Defines a start *Time codes* for the media.
- **end** (*string*) – Defines an end *Time codes* for the media.
- **template** (*string*) – Export template to use (see export-templates).
- **allowMissing** (*boolean*) –
  - `true` (default) - Job will be started and the missing files will be ignored.
  - `false` - Job will fail if there are missing files and the files could not be generated by transcoding. A shape tag should be specified.
- **track** (*string*) – Comma-separated list of item track ids to include as physical files. Can include wildcards, e.g. `A*`. Can also contain component ids. Since 4.17.8, can also be on the format `{shape}:track`, where shape is identified by shape id or shape tag. Default is `*`, all tracks/components.

  Since Vidispine 5.0, can also be of syntax: `shape:` {shape-id}:{track}, `cpl:` {CPL UUID}:{track}, `tag:` {shape-tag}:{track}, or `component:` {component-id}. (The `component:` syntax means the same as with no prefix.)
- **cplTrack** (*string*) – Comma-separated list of item track ids to include in the CPL. Can include wildcards, e.g. `A*`. Can also contain component ids. Since 4.17.8, can also be on the format `{shape}:track`, where shape is identified by shape id or shape tag. Default is the same as the track parameter.
- **version** (*string*) –
  - *essence-version-id* - Return shapes for a specified version.
  - `all` - Return shapes for all versions.
  - `latest` (default) - Return shapes for the latest version.
  - `latest-per-shapetag` - Return shapes with the highest essence version number per shape tag.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _export

### Start an export job for a single shape

**POST /item/** (*item-id*) **/shape/** *shape-id***/export** Creates a new export job that will copy a file from the specified shape to a remote location.

If the URI ends with a "/" the URI is assumed to describe a folder and the file will retain its existing filename. Otherwise it is assumed that the URI describes a file and that filename will be used.

**Query Parameters**

- **uri** (*string*) – A URI to the destination of the file.
- **locationName** (*string*) – The name of an *export location*.
- **metadata** (*boolean*) –
  - `true` - Metadata will also be exported to side-car XML file.
  - `false` (default) - No metadata is exported.
- **projection** (*string*) – Defines the projection to use when exporting the metadata.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **useOriginalFilename** (*boolean*) – If set to `true`, the file(s) will be exported with their original filename if available. Default is `false`.
- **useOriginalComponentFilename** (*boolean*) – If set to `true`, the file(s) will be exported with their original component filename if available.
- **start** (*string*) – Defines a start *time code* for the media.
- **end** (*string*) – Defines an end *time code* for the media.
- **allowMissing** (*boolean*) –
  - `true` (default) - Job will be started and the missing files will be ignored.
  - `false` - Job won't be started if there are missing files.
- **track** (*string*) – Comma-separated list of item track ids. Can include wildcards, e.g. `A*`. Can also contain component ids. Default is `*`, all tracks/components.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _export

### Start an export job for a collection or a library

**POST /collection/** (*collection-id*) **/export**
**POST /library/** (*library-id*) **/export**
Creates a new export job that will copy all matching files in the collection/library to a remote location.

A shape tag can be specified to decide which shapes that will be exported. The files will retain their original names and the URI should therefore point to the folder where the files should be placed.

**Query Parameters**

- **uri** (*string*) – A URI to the destination of the file.
- **locationName** (*string*) – The name of an *export location*.
- **metadata** (*boolean*) –
  - `true` - Metadata will also be exported to side-car XML file.
  - `false` (default) - No metadata is exported.
- **projection** (*string*) – Defines the projection to use when exporting the metadata.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **useOriginalFilename** (*boolean*) – If set to `true`, the file(s) will be exported with their original filename if available. Default is `false`.
- **useOriginalComponentFilename** (*boolean*) – If set to `true`, the file(s) will be exported with their original component filename if available.
- **tag** (*string*) – Finds a shape with the specified tag and uses that for export. If not specified, the system will attempt to use the original shape.
- **all** (*boolean*) –
  - `true` (default) - Fail the job if not all files from the selected shapes could be exported.
  - `false` - Don't export lost or unavailable files.
- **template** (*string*) – export template to use.
- **track** (*string*) – Comma-separated list of item track ids. Can include wildcards, e.g. `A*`. Can also contain component ids. Default is `*`, all tracks/components.
- **version** (*string*) –
  - *essence-version-id* - Return shapes for a specified version.
  - `all` - Return shapes for all versions.
  - `latest` (default) - Return shapes for the latest version.
  - `latest-per-shapetag` - Return shapes with the highest essence version number per shape tag.

**Produces**

- **application/xml, application/json** – JobDocument
- **application/xml, application/json** – JobDocument

**Role** _export

### Example

Create a new export job that transfers files in a certain collection that has shapes with the tag `flv`.

```
POST /collection/VX-10/export?tag=flv&uri=file:/home/user/video/
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-1334</jobId>
  <user>admin</user>
  <started>2010-05-24T14:53:12.732+02:00</started>
  <status>READY</status>
  <type>EXPORT</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

### Start an export job of an shape to an IMF package

**POST /item/** (*item-id*) **/shape/**
*shape-id***/export/imp** Creates a new export job that will create an IMF package as a remote location. URI must end with an "/" to denote a folder.

**Query Parameters**

- **uri** (*string*) – A URI to the destination of the file.
- **locationName** (*string*) – The name of an *export location*.
- **metadata** (*boolean*) –
  - `true` - Metadata will also be exported to side-car XML file.
  - `false` (default) - No metadata is exported.
- **projection** (*string*) – Defines the projection to use when exporting the metadata.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **useOriginalFilename** (*boolean*) – If set to `true`, the file(s) will be exported with their original filename if available. Default is `false`.
- **useOriginalComponentFilename** (*boolean*) – If set to `true`, the file(s) will be exported with their original component filename if available.
- **start** (*string*) – Defines a start *Time codes* for the media.
- **end** (*string*) – Defines an end *Time codes* for the media.
- **allowMissing** (*boolean*) –
  - `true` (default) - Job will be started and the missing files will be ignored.
  - `false` - Job won't be started if there are missing files.
- **track** (*string*) – Comma-separated list of item track ids. Can include wildcards, e.g. `A*`. Can also contain component ids. Default is `*`, all tracks/components.
- **cplTrack** (*string*) – Comma-separated list of item track ids to include in the CPL. Can include wildcards, e.g. `A*`. Can also contain component ids. Default is the same as the track parameter.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _export

### 17.11.2 Items

Manage items.

#### Managing items

#### List all items

**GET /item**

Returns a list of all items. This request is the same as performing an empty search.

Note that searching can also be performed by using the HTTP method PUT (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.6) using the same syntax, except for the parameter `q` is omitted and the ItemSearchDocument is sent in the body of the request.

**Content Parameters** See *Retrieving item information*

**Query Parameters**

- **result** (*string*) –
  - `list` (default) - Return a list of items.
  - `library` - Create a library with the matching items.
- **q** (*string*) – XML/JSON, ItemSearchDocument. Only with GET (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3).
- **library** (*string*) – Restricts search to within library, *Identifiers*. Default is `*`, all items.
- **first** (*integer*) – From the resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **cursor** (*string*) – New in version 4.16.
  - `*` - The initial cursor.
  - `string-from-search` - Cursor string returned from the search results.

  If set, the cursorMark (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#fetching-a-large-number-of-sorted-results-cursors) / search after (https://www.elastic.co/guide/en/elasticsearch/reference/6.8/search-request-search-after.html#search-request-search-after) features from Solr/Elasticsearch would be used to improve the deep paging (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#performance-problems-with-deep-paging) performance during a search.

  When `cursor` is used, The value of `first` will be ignored.

  Changed in version 5.5.

  Starting in 5.5, cursor is returned for the end of the result instead of null to enable tailing (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#tailing-a-cursor) search.
- **number** (*integer*) – The number of entities to fetch. Default is `100`.
- **count** (*boolean*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.

- **libraryId** (*string*) – If set, the library identified by this id will be used instead of creating a new library.
- **autoRefresh** (*boolean*) – When creating a library, make it *self-refresh*. Default is `false`.
- **updateMode** (*string*) – When creating a library, use this *update mode*. Default is `MERGE`.
- **updateFrequency** (*string*) – When creating a library, use this *update frequency*. Defaults to no periodic updates.
- **p** (*string*) – Comma-separated list of *paths* specifying the content to include. Overrides the content and filter parameters.
- **content** (*string*) – Comma-separated list of the types of content to retrieve. Valid values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **language** (*string*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **sampleRate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **track** (*string*) – Comma-separated list.

  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 – t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **include** (*string*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **includeValues** (*boolean*) – Return the value enumeration for each metadata field.
- **conflict** (*string*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **terse** (*string*) –
  - `yes` - Return metadata in *terse format*.
  - `no` (default) - Return metadata in verbose format.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** (*boolean*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.
- **revision** (*string*) – Specifying which metadata revision to display. Only used if requesting a single item or collection.
- **mergedType** (*string*) – The type of operation to check for.
- **mergedPermission** (*string*) – The lowest required permission level.
- **mergedExtradata** (*string*) – Any possible extra data.
- **uriType** (*string*) – Comma-separated list of format types (container format) to return.
- **scheme** (*string*) – URI scheme to return.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodType** (*string*) – *Access method*.
  - `AUTO` - Gives an APInoauth URI to the media. Access to file is tunneled through Vidispine.
  - `AZURE_SAS` - If the storage schema is azure:// you can get direct access to the media. The resulting URI will not tunnel through Vidispine but rather point directly to the media location at the azure storage.
- **methodMetadata** (*string*) – *Metadata* used with storage method.
- **tag** (*string*) – A *URI parameter*: Comma-separated list of *shape tags* to return.

- **version** (*string*) – Specifying which essence version to return for shapes. If special value `all`, display all versions. If special value `latest` (default), display latest version of shapes.
- **closedFiles** (*boolean*) – A *URI parameter*:
  - `true` (default) - Return only URIs that point to closed files.
  - `false` - Return all URIs.
- **storage** (*string[]*) – List of storage ids. Return only files from specific storages. Can be specified multiple times.
- **storageGroup** (*string*) – Storage group id. Return only files from storages specified in the storage group.
- **starttc** (*boolean*) –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.
- **url** (*boolean*) –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.
- **noauth-url** (*boolean*) –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **baseURI** (*string*) – Which base URI to use for the thumbnail URLs.
- **save** (*boolean*) –
  - `true` - Returns a `303 See Other`, with a `Location` header containing an URI to fetch the search result
  - `false` (default) - Returns a regular search result

**Produces**

- **application/xml, application/json** – ItemListDocument
- **text/plain** – CRLF-delimited list of ids or URLs

**Role** _item_search

**Role** _metadata_read (`content=metadata`)

**Role** _item_uri (`content=uri`)

**Role** _thumbnail_read (`content=poster` and `content=thumbnail`)

**Role** _access_control_read (`content=access` and `content=merged-access`)

**Role** _item_id_read (`content=external`)

**Tip:** Additional content can be retrieved by using the syntax specified in *Retrieving item information*.

### Retrieve an item

**GET /item/** (*item-id*)
Returns information about a single item.

**Query Parameters**

- **p** (*string*) – Comma-separated list of *paths* specifying the content to include. Overrides the content and filter parameters.
- **content** (*string*) – Comma-separated list of the types of content to retrieve. Valid values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **language** (*string*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **sampleRate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 − t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* * - Return metadata for all tracks of specified type, e.g. `A*`.

  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **include** (*string*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **includeValues** (*boolean*) – Return the value enumeration for each metadata field.
- **conflict** (*string*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **terse** (*string*) –
  - `yes` - Return metadata in *terse format*.
  - `no` (default) - Return metadata in verbose format.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** (*boolean*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.
- **revision** (*string*) – Specifying which metadata revision to display. Only used if requesting a single item or collection.
- **mergedType** (*string*) – The type of operation to check for.
- **mergedPermission** (*string*) – The lowest required permission level.
- **mergedExtradata** (*string*) – Any possible extra data.
- **uriType** (*string*) – Comma-separated list of format types (container format) to return.
- **scheme** (*string*) – URI scheme to return.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodType** (*string*) – *Access method*.
  - `AUTO` - Gives an APInoauth URI to the media. Access to file is tunneled through Vidispine.
  - `AZURE_SAS` - If the storage schema is azure:// you can get direct access to the media. The resulting URI will not tunnel through Vidispine but rather point directly to the media location at the azure storage.
- **methodMetadata** (*string*) – *Metadata* used with storage method.
- **tag** (*string*) – A *URI parameter*: Comma-separated list of *shape tags* to return.
- **version** (*string*) – Specifying which essence version to return for shapes. If special value `all`, display all versions. If special value `latest` (default), display latest version of shapes.
- **closedFiles** (*boolean*) – A *URI parameter*:
  - `true` (default) - Return only URIs that point to closed files.

- `false` - Return all URIs.

- **storage** (*string[]*) – List of storage ids. Return only files from specific storages. Can be specified multiple times.
- **storageGroup** (*string*) – Storage group id. Return only files from storages specified in the storage group.
- **starttc** (*boolean*) –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.
- **noauth-url** (*boolean*) –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **baseURI** (*string*) – Which base URI to use for the thumbnail URLs.

**Produces**

- **application/xml, application/json** – ItemDocument

**Role** _metadata_read (`content=metadata`)

**Role** _item_uri (`content=uri`)

**Role** _thumbnail_read (`content=poster` and `content=thumbnail`)

**Role** _access_control_read (`content=access` and `content=merged-access`)

**Role** _item_id_read (`content=external`)

---

**Tip:** Additional content can be retrieved by using the syntax specified in *Retrieving item information*.

---

### Delete an item

Deleting an item means removing the item's id, its metadata, shapes, and physical files.

**DELETE /item/** (*item-id*)

Marks the item as being deleted, meaning it will not be returned in search results. The actual removement from the database is done approximately once every minute. Also, all files associated with the item is marked as `TO_BE_DELETED`, meaning they will be deleted by the storage supervisor, but not sooner than all jobs involving the actual file has finished.

By specifying `keepShapeTagMedia` and/or `keepShapeTagStorage`, the files associated with the item is not deleted, but simply unassociated with the item.

If only `keepShapeTagMedia` is given, all files belonging to shapes of the item with any of the given shape tags are preserved.

If only `keepShapeTagStorage` is given, all files belonging to the item residing on the given storages are preserved. If both `keepShapeTagMedia` and `keepShapeTagStorage` is given, all files which *both* belongs to the specified shapes and storages are preserved.

If any of `keepShapeTagMedia` or `keepShapeTagStorage` contains a value `*`, then no files will be removed.

Changed in version 5.0: New _item_write role required.

**Query Parameters**

- **keepShapeTagMedia** (*string*) – Comma-separated list of shape tags whose files will not be deleted.
- **keepShapeTagStorage** (*string*) – Comma-separated list of storage ids whose files will not be deleted.

**Role** _item_wite

### Delete multiple items

Deleting an item means removing the item's id, its metadata, shapes, and physical files.

**DELETE /item**

Marks multiple items as being deleted, meaning they will not be returned in search results.

Changed in version 5.0: New _item_write role required.

**Query Parameters**

- **id** (*string*) – Comma-separated list of item ids or external ids. Must not be empty.
- **keepShapeTagMedia** (*string*) – Comma-separated list of shape tags whose files will not be deleted.
- **keepShapeTagStorage** (*string*) – Comma-separated list of storage ids whose files will not be deleted.

**Role** _item_write

### Search items

**PUT /item**

Performs an item search. If the `result` query parameter is set to `library` a new library is created, which can be used to further refine the search, using the `library` parameter.

**Content Parameters** See *Retrieving item information*

**Query Parameters**

- **result** (*string*) –
  - `list` (default) - Return a list of items.
  - `library` - Create a library with the matching items.
- **q** (*string*) – XML/JSON, ItemSearchDocument. Only with GET (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3).
- **library** (*string*) – Restricts search to within library, *Identifiers*. Default is `*`, all items.
- **first** (*integer*) – From the resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **cursor** (*string*) – New in version 4.16.
  - `*` - The initial cursor.
  - `string-from-search` - Cursor string returned from the search results.

  If set, the cursorMark (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#fetching-a-large-number-of-sorted-results-cursors) / search after (https://www.elastic.co/guide/en/elasticsearch/reference/6.8/search-request-search-after.html#search-request-search-after) features from Solr/Elasticsearch would be used to improve the deep paging (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#performance-problems-with-deep-paging) performance during a search.

When `cursor` is used, The value of `first` will be ignored.

Changed in version 5.5.

Starting in 5.5, cursor is returned for the end of the result instead of null to enable tailing (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#tailing-a-cursor) search.

- **number** (*integer*) – The number of entities to fetch. Default is `100`.
- **count** (*boolean*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **libraryId** (*string*) – If set, the library identified by this id will be used instead of creating a new library.
- **autoRefresh** (*boolean*) – When creating a library, make it *self-refresh*. Default is `false`.
- **updateMode** (*string*) – When creating a library, use this *update mode*. Default is `MERGE`.
- **updateFrequency** (*string*) – When creating a library, use this *update frequency*. Defaults to no periodic updates.
- **p** (*string*) – Comma-separated list of *paths* specifying the content to include. Overrides the content and filter parameters.
- **content** (*string*) – Comma-separated list of the types of content to retrieve. Valid values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.

- **language** (*string*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **sampleRate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 − t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **include** (*string*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **includeValues** (*boolean*) – Return the value enumeration for each metadata field.
- **conflict** (*string*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **terse** (*string*) –
  - `yes` - Return metadata in *terse format*.
  - `no` (default) - Return metadata in verbose format.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** (*boolean*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.
- **revision** (*string*) – Specifying which metadata revision to display. Only used if requesting a single item or collection.
- **mergedType** (*string*) – The type of operation to check for.
- **mergedPermission** (*string*) – The lowest required permission level.
- **mergedExtradata** (*string*) – Any possible extra data.
- **uriType** (*string*) – Comma-separated list of format types (container format) to return.
- **scheme** (*string*) – URI scheme to return.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodType** (*string*) – *Access method*.

  - `AUTO` - Gives an APInoauth URI to the media. Access to file is tunneled through Vidispine.
  - `AZURE_SAS` - If the storage schema is azure:// you can get direct access to the media. The resulting URI will not tunnel through Vidispine but rather point directly to the media location at the azure storage.
- **`methodMetadata`** (*`string`*) – *Metadata* used with storage method.
- **`tag`** (*`string`*) – A *URI parameter*: Comma-separated list of *shape tags* to return.
- **`version`** (*`string`*) – Specifying which essence version to return for shapes. If special value `all`, display all versions. If special value `latest` (default), display latest version of shapes.
- **`closedFiles`** (*`boolean`*) – A *URI parameter*:
  - `true` (default) - Return only URIs that point to closed files.
  - `false` - Return all URIs.
- **`storage`** (*`string[]`*) – List of storage ids. Return only files from specific storages. Can be specified multiple times.
- **`storageGroup`** (*`string`*) – Storage group id. Return only files from storages specified in the storage group.
- **`starttc`** (*`boolean`*) –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.
- **`url`** (*`boolean`*) –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.
- **`noauth-url`** (*`boolean`*) –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **`baseURI`** (*`string`*) – Which base URI to use for the thumbnail URLs.
- **`save`** (*`boolean`*) –
  - `true` - Returns a `303 See Other`, with a `Location` header containing an URI to fetch the search result
  - `false` (default) - Returns a regular search result

**Produces**

- **`application/xml, application/json`** – ItemListDocument
- **`text/plain`** – CRLF-delimited list of ids or URLs

**Accepts**

- **`application/xml, application/json`** – ItemSearchDocument

**Role** _item_search

**Role** _metadata_read (`content=metadata`)

**Role** _item_uri (`content=uri`)

**Role** _thumbnail_read (`content=poster` and `content=thumbnail`)

**Role** _access_control_read (`content=access` and `content=merged-access`)

**Role** _item_id_read (`content=external`)

---

**Tip:** There is a limit on how many items that can be returned for each call to this method. To get all items, iterate the calls, or even better in a batch scenario, start a job using *Listing items in batch* to get all items at once.

---

**Tip:** Additional content can be retrieved by using the syntax specified in *Retrieving item information*.

---

### Example

```
GET /item?result=library
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>product_category</name>
    <value>tv</value>
  </field>
</ItemSearchDocument>
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <library>VX*1233</library>
  <item>VY-1233</item>
  <item>VY-1234</item>
  <item>VX-7888</item>
</ItemListDocument>
```

```
PUT /item?library=VX*1233
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>created</name>
    <range>
      <value>2014-05-30T00:00:00+0200</value>
      <value>2014-06-03T07:30:00+0200</value>
    </range>
  </field>
</ItemSearchDocument>
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <library>VX*1234</library>
  <item>VY-1233</item>
  <item>VX-7888</item>
</ItemListDocument>
```

### Search history

### Retrieve search history

**GET /item/history**
Retrieves a list of searches made by a particular user, including "item search" and "Item and collection search". The results are ordered according to timestamp, with the latest searches being first. Duplicate queries will not be retrieved.

**Query Parameters**

- **start** (*string*) – An ISO 8601 date. If set, only searches made after this date will be retrieved.
- **maxResults** (*integer*) – The maximum number of searches that will be retrieved. The value must be between `1` and `50`. Default is `10`.
- **username** (*string*) – The name of the user that has performed the searched. If not specified, the user performing the request will be selected.

**Produces**

- **application/xml, application/json** – SearchHistoryDocument.

**Role** _item_search

### Re-index item

**PUT /item/** (*item-id*) **/re-index**
Queues a single item for re-index.

**Produces**

- **text/plain** –

See *Re-indexing metadata* if you wish to reindex all items in the system.

### Listing items in batch

### Create an item list job

**POST /item/list**
Starts a new job that goes through all the items available to the user/group and outputs a file to the supplied URI.

If no user and no group is supplied, all items will be retrieved. The output format depends on the specified parameter, if set to XML an ItemListDocument will be produced. Furthermore if an XSLT is given the ItemListDocument will be transformed.

**Query Parameters**

- **destinationUri** (*string*) – Required. The URI to output the CSV file to.
- **username** (*string*) – Filter items according to the access of the specified user.
- **groupname** (*string*) – Filter items according to the access of the specified group.
- **field** (*string*) – Comma-separated list of metadata fields to include in the result. Default is `title`
- **outputFormat** (*string*) – Specifies the output format. One of `xml` (default) and `csv`.
- **data** (*string*) – Specifies any additional data that should be included with the metadata fields.
- **p** (*string*) – Comma-separated list of *paths* specifying the content to include. Overrides the field and data parameters. Only supported for XML output.

- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Accepts**

- **application/xslt** – An optional XSLT capable of transforming ItemListDocument.

**Produces**

- **application/xml, application/json** – JobDocument.

**Role** _administrator

### Example

```
POST /item/list?field=title,durationSeconds&username=admin&destinationUri=file:/home/
 ↪user/output.csv&outputFormat=csv
Content-Type: application/xml

<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-64</jobId>
  <user>admin</user>
  <started>2010-11-29T11:12:55.768+01:00</started>
  <status>READY</status>
  <type>LIST_ITEMS</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

```
$ cat /home/user/output.csv
"itemId","format","fileSize","downloads","metadataField-title","metadataField-
 ↪durationSeconds"
"VX-22","mxf","10000000","1","","180.0"
"VX-18","mp3,aac","5876698,4253659","0","","212.242695"
"VX-12","flv","23939202","5","This is ""the"" title.","142.124698"
"VX-8","flv","5684452","3","","12.412487"
```

## Parent collections

### List collections that contain an item

**GET /item/** (*item-id*) **/collections**

Retrieves the ids of all collections that includes the item, and that the calling user has read access to.

**Produces**

- **application/xml, application/json** – URIListDocument containing the collection ids
- **text/plain** – CRLF-delimited list of ids

### Example

```
GET /item/VX-94/collections
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>VX-23</uri>
  <uri>VX-64</uri>
</URIListDocument>
```

#### Parent Libraries

#### List libraries that contain an item

New in version 4.16.1.

**GET /item/** (*item-id*) **/library**
Retrieves the ids of all libraries that includes the item, and that the calling user has read access to.

---

**Note:** This endpoint will not return any transient libraries.

---

**Produces**

- **application/xml, application/json** – URIListDocument containing the library ids
- **text/plain** – CRLF-delimited list of ids

### 17.11.3 Retrieving item information

Item content can be retrieved from different resources, the query parameters used are the same for the different resources. Below a table of the different supported resources can be seen.

| Name | BASE_PATH |
|---|---|
| Search | `/item`, `/search` |
| Specific items | `/item/{item-id}` |
| Libraries | `/library/{library-id}` |
| Collections | `/collection/{collection-id}/item` |

#### Get item information

By using a content parameter, much information can be gathered in one single call to the API.

#### Get information

**GET {item-content-resource}**
Retrieves the types of content that are specified in `content`. If URIs are included then the parameters `type` or `tag` needs to be set.

**Query Parameters**

- **includeConstraintValue** (*string*) – Comma-separated list of fields whose "display value" should be retrieved from the *metadata dataset*.
  - `all` (default) - Return the "display value" of all fields.
  - `none` - No "display value" will be returned. The fields will only have `id` set.
  - *comma-separated field names* - Return the "display value" of the specified fields.
- **p** (*string*) – Comma separated list of *paths* specifying the content to include. Overrides the content and filter parameters.

- **content** (*string*) – Comma-separated list of the types of content to retrieve. Valid values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** (*string*) – A metadata parameter, see *Retrieve metadata*.
- **field** (*string*) – A metadata parameter, see *Retrieve metadata*.
- **group** (*string*) – A metadata parameter, see *Retrieve metadata*.
- **language** (*string*) – A metadata parameter, see *Retrieve metadata*.
- **sampleRate** (*string*) – A metadata parameter, see *Retrieve metadata*.
- **track** (*string*) – A metadata parameter, see *Retrieve metadata*.
- **terse** (*boolean*) – A metadata parameter, see *Retrieve metadata*.
- **include** (*string*) – A metadata parameter, see *Retrieve metadata*.
- **type** (*string*) – A *URI parameter*: Comma-separated list of format types (container format) to return.
- **tag** (*string*) – A *URI parameter*: Comma-separated list of *shape tags* to return.
- **scheme** (*string*) – A *URI parameter*: URI scheme to return, e.g. `ftp`.
- **closedFiles** (*boolean*) – A *URI parameter*:
  - `true` (default) - Return only URIs that point to closed files.
  - `false` - Return all URIs.
- **noauth-url** (*boolean*) – If `true`, thumbnail URIs that do not need authentication are returned. If `false` (default), normal thumbnail URIs are returned.
- **defaultValue** (*boolean*) – A metadata parameter, see *Retrieve metadata*.
- **methodType** (*string*) – Type of storage method. When returning URIs, only use methods of this type. See *Storages*.
- **methodMetadata** (*string[]*) – Metadata used with storage method. See *Storages*.
- **version** (*string*) – Specifying which essence version to return for shapes. If special value `all`, display all versions. If special value `latest` (default), display latest version of shapes. If special value `latest-per-shapetag`, display shapes with the highest essence version number per shape tag.
- **revision** (*string*) – Specifying which metadata to display. Only used if requesting a single item or collection.

**Produces**

- **application/xml, application/json** – ItemDocument

**Role** _metadata_read (`content=metadata`)

**Role** _item_uri (`content=uri`)

**Role** _thumbnail_read (`content=poster` and `content=thumbnail`)

**Role** _access_control_read (`content=access` and `content=merged-access`)

**Role** _item_id_read (`content=external`)

### Example

Retrieving terse metadata and thumbnails for an item.

```
GET /API/item/VX-123/?content=metadata,thumbnail&terse=yes
```

```
<ItemDocument id="VX-123">
    <thumbnails>
        <uri>http://example.com/API/thumbnail/VX-1/VX-123/0@1000000</uri>
        <uri>http://example.com/API/thumbnail/VX-1/VX-123/1000000@1000000</uri>
        <uri>http://example.com/API/thumbnail/VX-1/VX-123/2000000@1000000</uri>
    </thumbnails>
    <terse>
        <durationSeconds end="+INF" start="-INF">2.04</durationSeconds>
        <durationTimeCode end="+INF" start="-INF">2040000@1000000</durationTimeCode>
        <field_A end="7" start="3">ABC</field_A>
        <title end="+INF" start="-INF">This is an imported item!</title>
        <user end="+INF" start="-INF">testUser</user>
    </terse>
</ItemDocument>
```

### Get item content in the search result

The parameters above can also be used when searching (*Search*). Note that only content the user has sufficient permissions for will be retrieved.

### Example

Retrieving the URIs to all AVI containers that can be accessed either by HTTP or FTP for all items.

```
GET /API/item/?content=uri&type=avi&scheme=http,ftp
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <item id="VX-123">
        <files>
            <uri>ftp://example.com/VX-123_VX-2189.avi</uri>
        </files>
        <timespan start="-INF" end="+INF"/>
    </item>
    <item id="VX-124">
        <files/>
        <timespan start="-INF" end="+INF"/>
    </item>
    <item id="VX-125">
        <files>
            <uri>http://example.com/VX-125_VX-3180.avi</uri>
            <uri>ftp://example.com/VX-125_VX-3180.avi</uri>
        </files>
        <timespan start="-INF" end="+INF"/>
    </item>
</ItemListDocument>
```

### Retrieving URIs to the content of an item

The URI retrieval method is a scaled-down version of the *Get information* method above.

### Get item URI

**GET /item/** (*item-id*) **/uri**

Retrieves the URI to any container contained in the item that matches the specified type or the files contained in a shape that matches the given tags.

**Query Parameters**

- **type** (*string*) – Comma-separated list of format types (container format) to return.
- **tag** (*string*) – Comma-separated list of *shape tags* to return.
- **scheme** (*string*) – URI scheme to return, e.g. `ftp`.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodType** (*string*) – *Access method*.
  - `AUTO` - Gives an APInoauth URI to the media. Access to file is tunneled through Vidispine.
  - `AZURE_SAS` - If the storage schema is azure:// you can get direct access to the media. The resulting URI will not tunnel through Vidispine but rather point directly to the media location at the azure storage.
- **methodMetadata** (*string*) – *Metadata* used with storage method.
- **closedFiles** (*boolean*) –
  - `true` (default) - Return only URIs that point to closed files.
  - `false` - Return all URIs.

**Produces**

- **application/xml, application/json** – URIListDocument

**Role** _item_uri

### Example

```
GET /item/VX-123/uri?type=avi&tag=lowres
Accept: application/xml
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <uri>http://example.com/VX-123_VX-5003.avi</uri>
    <uri>ftp://user:password@example.com/VX-123_VX-5003.avi</uri>
</URIListDocument>
```

## 17.11.4 Item locks

Items can be locked by users to temporarily prevent access from other users. This can be used to prevent users from working with stale and conflicting data. Locks should not be seen as an alternative to access control, as any user that has write access to an item can remove the locks.

If any user attempts to access an item that is locked by another user, HTTP status code 409 Conflict (http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.4.10) will be returned. For example:

```
HTTP/1.1 409 Operation would lead to conflict
Context: lock
ID: VX-123
```

```
Reason: That entity is locked by another user.
Value: the-name-of-the-other-user
```

### Managing locks

All locks are associated with an expiration date and will be removed after they expire.

### Create a lock

**POST /item/** (*item-id*) **/lock**

Creates a new lock for the item with an expiration date. The expiration date is the sum of the timestamp and the duration. If no timestamp and no duration is given, the expiration date will be set to 24 hours forward in time.

**Query Parameters**

- **timestamp** (*string*) – An ISO 8601 timestamp. Defaults to the current time.
- **duration** (*string*) – An ISO 8601 duration. Default is `0`.

**Status Codes**

- **200 OK** – The lock was created.
- **409 Conflict** – Some other user already holds a lock on that item.

**Role** _lock_write

### Example

Create a lock for a specific timestamp:

```
POST /item/VX-123/lock?timestamp=2010-08-20T15:00:00+02:00
```

```
200 OK
```

Create a lock for 3 hours:

```
POST /item/VX-123/lock?duration=PT3H
```

```
200 OK
```

### Retrieve a lock

**GET /item/** (*item-id*) **/lock**

Retrieves information about the expiration date and which user that holds the lock.

**Status Codes**

- **404 Not Found** – Either the item or the lock could not be found.

**Produces**

- **application/xml, application/json** – LockDocument

**Role** _lock_read

#### Example

```
GET /item/VX-123/lock
```

```
<LockDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <id>VX-123</id>
   <user>admin</user>
   <expires>2010-08-20T15:00:00.000+02:00</expires>
</LockDocument>
```

### Delete a lock

**DELETE /item/** (*item-id*) **/lock**

Removes the lock for the item.

**Status Codes**

- **200 OK** – The lock was removed.

**Role** _lock_write

#### Example

```
DELETE /item/VX-123/lock
```

```
200 OK
```

### Extend the expiration date of a lock

**PUT /item/** (*item-id*) **/lock**

Sets a new expiration date for the lock. The expiration date is the sum of the timestamp and the duration. If no timestamp and no duration is given, the expiration date will be set to 24 hours forward in time.

**Query Parameters**

- **timestamp** (*string*) – An ISO 8601 timestamp. Defaults to the current time.
- **duration** (*string*) – An ISO 8601 duration. Default is `0`.

**Status Codes**

- **200 OK** – The lock was extended.

**Role** _lock_write

#### Example

```
POST /item/VX-123/lock?timestamp=2010-08-20T16:00:00+02:00
```

```
200 OK
```

## 17.11.5 Item-to-item relations

This section describes relations between items. The relation can be used to find ancestors, derived items, or simply loosely related items.

### Type of relations

Relations

- can be directional or undirectional. In a directional relation, one item is the source and another item is the target. In an undirectional relation, the two items are treated equally
- are manually built using the API or created automatically. An example of automatically built relations is the timeline conform method, which automatically creates directed relations
- have metadata as key-value pairs. One key-value pair which is always present is the `type` key, which describes the reason of the relationship.

### Automatically generated relations

Item-to-item relations are automatically generated by timeline conform actions. These relations are directional from source item(s) to target item. The relations have the following tags:

- `key=conform`
- `conform-job=` { **conform-job** }

### Managing item relations

### List all item relations

**GET /item/** (*id*) **/relation**

Returns a list of relations that matches the search criteria. Item id can be an *Identifiers*, that is libraries can be used.

**Query Parameters**

- **direction** (*string*) –
  - `U` - Only return undirectional relations where `id` is part of.
  - `S` - Only return directional relations where `id` is the source item.
  - `T` - Only return directional relations where `id` is the target item.
  - `D` - Only return directional relations where `id` is the source or target item.
  - `A` (default) - Return all relations that `id` is a part of.

**Status Codes**

- **400** – An invalid direction has been specified.
- **404** – Could not find the item identified by `id`.

**Produces**

- **application/xml, application/json** – ItemRelationListDocument.
- **text/plain** – *CR LF* -delimited list of *Tabbed tuples* of relation id, relation URI, direction type (`U`, `D`), relation type, and source id, target id.

**Role** _relation_read

In addition, extra query parameters of the form `key=value` can be added, to only return relations that matches the key-value pair(s).

### Create an item relation

**POST /item/**(*id1*)**/relation/**

*id2* Generates a new relation between the two items with the given ids, `id1` and `id2`, with the given parameters.

**Query Parameters**

- **direction** (*string*) – Required.
  - `U` - Set the direction of the relation as undirectional.
  - `S` - Set the direction as `id1` being the source and `id2` being the target.
  - `T` - Set the direction as `id2` being the source and `id1` being the target.
- **allowDuplicate** (*boolean*) –
  - `true` (default) - Allow duplicate relations.
  - `false` - Avoid adding duplicate relations.

**Status Codes**

- **400 Bad request** – Both `id1` and `id2` identifies the same item, or the direction is invalid.
- **404 Not found** – Could not find the item identified by `id1` or `id2`.

**Produces**

- **application/xml, application/json** – ItemRelationDocument

**Role** _relation_write

In addition, extra query parameters of the form `key=value` can be added, to set metadata of the item-to-item relation.

### Create multiple item relations

New in version 4.16.

**POST /relation**

Generates multiple relations at once. Each relation has a `source` and a `target`, and the direction can take the value `U`, if not set it generates a directional relation from `source` to `target`.

**Query Parameters**

- **allowDuplicate** (*boolean*) –
  - `true` (default) - Allow duplicate relations.
  - `false` - Avoid adding duplicate relations.

**Status Codes**

- **400 Bad request** – Both `source` and `target` identifies the same item, or the direction is invalid.
- **404 Not found** – Could not find the item identified by `source` or `target`.

**Accepts**

- **application/xml, application/json** – ItemRelationListDocument

**Produces**

- **application/xml, application/json** – ItemRelationListDocument

**Role** _relation_write

For example:

```
<?xml version="1.0"?>
<ItemRelationListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <relation>
    <direction>
      <source>VX-1</source>
      <target>VX-2</target>
    </direction>
  </relation>
  <relation>
    <direction>
      <source>VX-1</source>
      <target>VX-3</target>
    </direction>
  </relation>
  <relation>
      <direction type="U">
        <source>VX-4</source>
        <target>VX-5</target>
      </direction>
   </relation>
</ItemRelationListDocument>
```

### Retrieve an item relation

**GET /relation/** (*relation-id*)
Retrieves the relation with the id `relation-id`.

**Status Codes**

- **404 Not found** – Could not find the relation identified by `relation-id`.

**Produces**

- **application/xml, application/json** – ItemRelationDocument.

**Role** _relation_read

### Update an item relation

**PUT /relation/** (*relation-id*)
Updates the relation metadata for a relation with the id `relation-id`.

**Query Parameters**

- **direction** (*string*) –
  - `U` - Set the direction of the relation as undirectional.
  - `S` - Set the direction as `id1` being the source and `id2` being the target.
  - `T` - Set the direction as `id2` being the source and `id1` being the target.

**Status Codes**

- **404 Not found** – Could not find the relation identified by `relation-id`.

**Produces**

- **application/xml, application/json** – The updated item described as an ItemRelationDocument.

**Role** _relation_write

Query parameters of the form `key=value` are used to modify the metadata of the relation.

### Delete an item relation

**DELETE /relation/**(*relation-id*)
Deletes the relation with the id `relation-id`.

**Status Codes**

- **200 OK** – The item relation is deleted.
- **404 Not found** – Could not find the relation identified by `relation-id`.

**Role** _relation_write

### Delete all item relations

**DELETE /item/**(*id*)**/relation**
Deletes the relations with the specified direction or all relations.

**Query Parameters**

- **direction** (*string*) –
  - `A` - This is the default value. Deletes all relations `id1` is involved in.
  - `U` - Deletes only the relations with the direction as undirectional.
  - `S` - Deletes only the relations where `id1` is the source and `id2` is the target.
  - `T` - Deletes only the relations where `id2` is the source and `id1` is the target.

**Status Codes**

- **200 OK** – The item relation is deleted.

**Role** _relation_write

### Delete all relations between two items

**DELETE /item/**(*id1*)**/relation/**
*id2* Deletes the relations with the specified direction or all relations between `id1` and `id2`.

**Query Parameters**

- **direction** (*string*) –
  - `A` - This is the default value. Deletes all relations between `id1` and `id2`.
  - `U` - Deletes only the relations with the direction as undirectional.
  - `S` - Deletes only the relations where `id1` is the source and `id2` is the target.
  - `T` - Deletes only the relations where `id2` is the source and `id1` is the target.

**Status Codes**

- **200 OK** – The item relation is deleted.

**Role** _relation_write

### 17.11.6 Item sequences

A sequence is an assembly of audio and video from other items. An item may have multiple sequences, but they will all be considered equivalent by Vidispine, that is, that they represent the same logical sequence.

Sequences can be *imported and exported* to and from common NLE formats.

#### Rendering a sequence

A sequence can be rendered which creates a new shape that for example can be used as a preview of the sequence. The shape tag that is provided must have a transcode preset the specifies at least:

- The container format.
- The audio codec and bitrate (optional for PCM.)
- The video codec and bitrate.

The transcoder can render a subset of the effects (both normal and key framed) and transitions that are available in Final Cut and Avid Media Composer. They are:

- Effects
  - Crop
  - Position
  - Scale
  - Rotate
  - Opacity
- Transitions
  - Dissolves
    - Cross dissolve
    - Dither dissolve
    - Fade in fade out dissolve
  - Wipes
    - Band wipe
    - Centre wipe
    - Checker wipe
    - Inset wipe
  - Iris wipes
    - Cross iris
    - Diamond iris
    - Oval iris
    - Rectangle iris
    - Star iris

### Sequence operations

### List all sequences

**GET /item/** (*id*) **/sequence**
Retrieves the sequences that have been stored for a specific item.

**Status Codes**

- **404 Not found** – Could not find the item

**Produces**

- **application/xml, application/json** – SequenceListDocument

**Role** _sequence_read

### Update or create a sequence

**PUT /item/** (*id*) **/sequence/**
*format* Creates or updates the sequence in the given format.

**Query Parameters**

- **pauseFrame** (*integer*) – When a rendering job is started, this parameter determines which frame the job will pause at. The job will resume when the sequence is updated.

**Status Codes**

- **404 Not found** – Could not find the item

**Accepts**

- **application/octet-stream** – The sequence definition

**Produces**

- **application/xml, application/json** – ItemDocument with the id of the sequence

**Role** _sequence_write

### Retrieve a sequence

**GET /item/** (*id*) **/sequence/**
*format* Retrieves the definition of the sequence in the given format.

**Status Codes**

- **404 Not found** – Could not find the item

**Produces**

- ***/*** – Media type based on the format.

**Role** _sequence_read

### Delete a sequence

**DELETE /item/** (*id*) **/sequence/**
*format* Removes a specific sequence from an item.

**Status Codes**

- **404 Not found** – Could not find the item

- **404 Not found** – Could not find the sequence

**Role** _sequence_write

### Conform metadata

**POST /item/** (*id*) **/sequence/conform-metadata**

Updates the item metadata with the metadata from the items listed in the sequence. The metadata will be selected based on the intervals.

**Role** _metadata_write

### Render a standalone sequence

**POST /sequence/render**

Creates a new job that renders the given sequence. A new item will be created containing a shape with the rendered result once the job is finished.

**Query Parameters**

- **tag** (*string[]*) – The shape tag specifying the format of the rendered sequence.
- **sourceTag** (*string[]*) – The shape tag specifying the shapes to use as input.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **destinationItem** (*string*) – An item id, to which the new new shape will be associated.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.

  New in version 4.16.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM`.
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Status Codes**

- **404 Not found** – Could not find the item

**Accepts**

- **application/xml, application/json** – SequenceRenderRequestDocument

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

### Render a sequence

**POST /item/** (*id*) **/sequence/render**

Creates a new job that renders the sequence for the given item. The item will contain a new shape with the rendered result once the job is finished.

**Query Parameters**

- **tag** (*string[]*) – The shape tag specifying the format of the rendered sequence.
- **sourceTag** (*string[]*) – The shape tag specifying the shapes to use as input.
- **subtitleLanguage** (*string*) – The language code specifying the *subtitle* language to use.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **destinationItem** (*string*) – An item id, to which the new new shape will be associated.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.

  New in version 4.16.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM`.
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Status Codes**

- **404 Not found** – Could not find the item

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _sequence_read

**Role** _job_write

### Example

```
POST /item/VX-8/sequence/render?tag=h264
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-13</jobId>
  <user>admin</user>
  <started>2011-10-26T20:23:11.897Z</started>
  <status>READY</status>
  <type>CONFORM</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

## 17.11.7 Shapes

Manage shapes for an item.

### Item shapes

### List all shapes

**GET /item/** (*id*) **/shape**
Returns all existing shapes for a specified item.

**Query Parameters**

- **version** (*string*) –
  - *essence-version-id* - Return shapes for a specified version.
  - `all` - Return shapes for all versions.
  - `latest` (default) - Return shapes for the latest version.
  - `latest-per-shapetag` - Return shapes with the highest essence version number per shape tag.
- **tag** (*string*) – Comma-separated list. Only return shapes with these tags.
- **url** (*boolean*) –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.
- **placeholder** (*string*) –
  - `true` - Only return placeholder shapes.
  - `false` (default) - Only return non-placeholder shapes.
  - `all` - Return all shapes.

**Status Codes**

- **404 Not found** – Invalid id

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – CRLF-delimited list of ids or URLs

**Role** _item_shape_read

### Retrieve a shape

**GET /item/** (*id*) **/shape/**
*shape-id* Returns a shape for a specified item.

**Query Parameters**

- **methodType** (*string*) – Return URIs from storage methods with a particular *type*. By default, return URLs with empty `methodType`.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodMetadata** (*string[]*) – *metadata* used with storage method.
- **scheme** (*string*) – URI scheme to return.
- **transient** (*boolean*) –
  - `true` - Return the shape by inspecting the file on disk. Use with growing files to get an as up-to-date shape as possible.
  - `false` (default) - Return the shape that was last read from the file.
- **includePlaceholder** (*boolean*) –
  - `true` - Include expected but not yet imported components in the shape.
  - `false` (default) - Do not include placeholder components.

**Status Codes**

- **404 Not found** – Invalid id

**Produces**

- **application/xml, application/json** – ShapeDocument

**Role** _item_shape_read

### Retrieve a graphical representation of a shape

**GET /item/**(*id*)**/shape/***shape-id***/graph** Shows components and tracks in a graphical format.

**Produces**

- **image/png** –

**Role** _item_shape_read

### Retrieve a shape as a dot file

**GET /item/**(*id*)**/shape/***shape-id***/graph/dot** Shows components and tracks in a graphical format in dot format, for further processing.

**Produces**

- **text/plain** –

**Role** _item_shape_read

### Retrieve a shape as an IMF CPL

**GET /item/**(*id*)**/shape/***shape-id***/cpl** Returns component information as CPL.

**Produces**

- **application/xml** –

**Role** _item_shape_read

### Delete a shape

**DELETE /item/**(*id*)**/shape/***shape-id* Removes the specified shape. This will remove all components and and mark files for deletion, unless files are used in other shapes.

**Query Parameters**

- **url** (*boolean*) –
  - `true` - Instead of shape ids, return the full paths of the shapes in the response document.
  - `false` (default) - Only return the ids of the remaining shapes.
- **keepFiles** (*boolean*) –
  - `true` - Keep the files belong to this shape.
  - `false` (default) - Remove the files belong to this shape.
- **updateMetadata** (*boolean*) –

- `true` - Remove the item metadata that is generate from this shape
- `false` (default) - Keep the item metadata that is generate from this shape

**Produces**

- **`application/xml, application/json`** – URIListDocument
- **`text/plain`** – CRLF-delimited list of ids or URLs

**Role** _item_shape_write

### Delete all shapes

**`DELETE /item/`**(*id*)**`/shape/`**

Removes all shapes, regardless of essence version, for the specified item. This will remove all components and mark files for deletion, unless files are used in other shapes.

To delete all shapes for a specific essence version, see *`DELETE /item/(id)/shape/version/(version)`*.

**Query Parameters**

- **`keepFiles`** (*`boolean`*) –
  - `true` - Keep the files belong to the shapes.
  - `false` (default) - Remove the files belong to the shapes.

**Role** _item_shape_write

### Importing a new shape

New shape can be imported in one of two methods. Both methods share a lot of similarities to item imports, *using a URI* or *using the request body*. The difference between a shape import and an essence version import is that it does not increment the essence version nor does it perform any transcoding.

### Import a shape using a URI or an existing file

**`POST /item/`**(*id*)**`/shape`**

Starts a new shape import job using either a URI or a file id.

**Query Parameters**

- **`uri`** (*`string`*) – A URI to the file that will be imported. Make sure to *percent encode* the URI. Must be specified unless `fileId` is specified.
- **`fileId`** (*`string`*) – The id of the file that contains the essence. Must be specified unless `uri` is specified.
- **`allowReimport`** (*`boolean`*) –
  - `true` - Import the file to this shape even if the file is already importing or is already part of another item.
  - `false` (default) Reject the request if the file with the given id has already been imported or is currently importing.
- **`tag`** (*`string`*) – The tags to assign to the new shape.
- **`storageId`** (*`string`*) – Identifier of storage where essence file is to be stored.
- **`notification`** (*`string`*) – The *placeholder job notification* to use for this job.

- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

### Import a shape using the request body

**POST /item/** (*id*) **/shape/raw**

Starts a new shape import job using the data in the request data.

**Query Parameters**

- **tag** (*string*) – The tags to assign to the new shape.
- **storageId** (*string*) – Identifier of storage where essence file is to be stored.
- **filename** (*string*) – The filename to be stored as original filename
- **transferPriority** (*integer*) – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers. A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **transferId** (*string*) – An id to assign the transfer to be able to refer to it.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Status Codes**

- **400** – If the amount of data received does not match the given Content-Length header.

**Accepts**

- **application/octet-stream** – The raw essence data.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

### Create a shape using shape technical information

**POST /item/** (*id*) **/shape/create**

Creates a new shape using the supplied information.

Changed in version 4.17.2: If the shape document has components that reference files (by id), then these files will be associated with the corresponding component.

**Query Parameters**

- **tag** (*string*) – The tags to assign to the new shape.
- **updateItemMetadata** (*boolean*) – If the shape is tagged `original` and this query parameter is `true`, the item's system metadata (e.g. `durationSeconds`) is updated. Default is `false`.

**Accepts**

- **application/xml, application/json** – ShapeDocument

**Produces**

- **application/xml, application/json** – ShapeDocument

**Role** _import

### Import a shape from an IMF package

**POST /item/** (*id*) **/shape/imp**

Starts a new shape import job using a URI of an IMF asset map

Changed in version 5.3: IMF packages will now be validated using Photon and results saved as metadata on the item. Can be disabled with **jobMetadata** parameter.

**Query Parameters**

- **uri** (*string*) – The URI of the asset map
- **tag** (*string*) – The tags to assign to the new shape.
- **storageId** (*string*) – Identifier of storage where essence file is to be stored.
- **removeOldEssenceFiles** (*boolean*) –
  - `true` - Remove files associated with shapes with same tags and lower essence version.
  - `false` (default) - Keep the files belong to the shapes.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **importTag** (*string[]*) – A list of shape tags that the created shape will be associated with. Default is `original`.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

The IMF shape import job accepts certain special **jobMetadata** parameters:

**skipImpValidation** If set to true, do not perform Photon IMF package validation.

New in version 5.3.

### Creating thumbnails and posters

Thumbnails and posters of a specific shape can be created by starting a thumbnail job.

### Start a thumbnail job

**POST /item/** (*item-id*) **/shape/**
*shape-id***/thumbnail** Creates a new thumbnail job with the specified parameters. Note that a job cannot both create thumbnails at specified intervals and posters. Creating thumbnails according to transcoder rules and creating posters is however allowed.

Changed in version 5.0: For multi-layer PSD/PSB files, only a thumbnail of all layers flattened will be generated by default.

**Query Parameters**

- **createThumbnails** (*boolean*) –
  - `true` - Creates thumbnails according to default transcoder rules.
  - *t1*, ... - Thumbnails will be created on the specified, comma-separated, *time codes*.
  - `false` (default) - No thumbnails will be created.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **thumbnailWidth** (*integer*) – The width of the thumbnails. If `thumbnailWidth` is specified, `thumbnailHeight` must also be specified.
- **thumbnailHeight** (*integer*) – The height of the thumbnails. If `thumbnailHeight` is specified, `thumbnailWidth` must also be specified.
- **thumbnailPeriod** (*string*) – Timecode string specifying the interval of the thumbnails. It should be a decimal integer when working with multi-page images/PDFs, meaning every N page(s).
- **posterWidth** (*integer*) – The width of the posters.
- **posterHeight** (*integer*) – The height of the posters.
- **posterFormat** (*string*) –
  - `jpeg` (default) - Creates posters in JPEG format.
  - `png` - Creates posters in PNG format.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **tag** (*string*) – Include additional video settings from this transcode preset. Resolution settings in the tag are overridden by query parameters `thumbnailHeight` and `thumbnailWidth`.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.

**Produces**

- **application/xml, application/json** – JobDocument

### Essence versions

New versions of essence can be imported in one of two methods. Both methods share a lot of similarities to item imports, *using a URI* or *using the request body*.

### List all essence versions

**GET /item/** (*id*) **/shape/version**
Returns a list containing URLs to all essence versions of the item.

**Produces**

- **application/xml, application/json** – *EssenceVersionListDocument* containing information to all essence versions of the item.

**Role** _item_shape_read

### Import an essence version using a URI or an existing file

**POST /item/** (*id*) **/shape/essence**
Starts a new essence import job using either a URI or a file id.

**Query Parameters**

- **uri** (*string*) – A URI to the file that will be imported. Make sure to *percent encode* the URI. Must be specified unless `fileId` is specified.
- **fileId** (*string*) – The id of the file that contains the essence. Must be specified unless `uri` is specified.
- **allowReimport** (*boolean*) –
  - `true` - Import the file to this shape even if the file is already importing or is already part of another item.
  - `false` (default) Reject the request if the file with the given id has already been imported or is currently importing.
- **storageId** (*string*) – Identifier of storage where essence file is to be stored.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **tag** (*string*) – The tags to assign to the new shape.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

### Import an essence version using the request body

**POST /item/** (*id*) **/shape/essence/raw**

Starts a new essence import job using the data in the request data.

**Query Parameters**

- **tag** (*string*) – The tags to assign to the new shape.
- **storageId** (*string*) – Identifier of storage where essence file is to be stored.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **filename** (*string*) – The filename to be stored as original filename
- **transferPriority** (*integer*) – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers. A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **transferId** (*string*) – An id to assign the transfer to be able to refer to it.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Status Codes**

- **400** – If the amount of data received does not match the given Content-Length header.

**Accepts**

- **application/octet-stream** – The raw essence data.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

### Retrieve an essence version

**GET /item/** (*id*) **/shape/version/**

*version* Returns a list of shapes from the specified version.

**Produces**

- **application/xml, application/json** – *EssenceVersionDocument* containing all the shapes with the specified version.

**Role** _item_shape_read

### Delete an essence version

**DELETE /item/** (*id*) **/shape/version/**

*version* Deletes all shapes associated with the specified version. Thumbnails connected to the version will also be deleted.

**Role** _item_shape_write

### Copy an essence version of a shape to a new version

**POST /item/** (*id*) **/shape/**
*shape-id***/version** Copies the specified shape to a new shape, with the new latest essence version number.

**Produces**

- **application/xml, application/json** – ShapeDocument containing the new shape.

**Role** _item_shape_write

### Copy an essence version of a shape to a specific version

**PUT /item/** (*id*) **/shape/**
*shape-id***/version/***new-version* Copies the specified shape to a new shape, with the given essence version number.

**Produces**

- **application/xml, application/json** – ShapeDocument containing the new shape.

**Role** _item_shape_write

### Placeholder shapes

### Create a placeholder shape

**POST /item/** (*id*) **/shape/placeholder**
Creates an new placeholder shape for a specific item.

**Query Parameters**

- **tag** (*string*) – Comma-separated shape tags to be added to the shape.
- **container** (*integer*) – The number of container components
- **audio** (*integer*) – The number of audio components
- **video** (*integer*) – The number of video components
- **binary** (*integer*) – The number of binary components
- **frameDuration** (*string*) – *duration* for each image in the sequence.

**Accepts**

- **application/xml, application/json** – SimpleMetadataDocument

**Produces**

- **text/plain** – The id of the new shape.

**Role** _import

### Update a placeholder shape

**PUT /item/** (*id*) **/shape/**
*shape-id***/placeholder** Updates the expected number of container, video, audio and binary components for a specific placeholder shape.

**Query Parameters**

- **tag** (*string*) – Comma-separated shape tags to be added to the shape.

- **container** (*integer*) – The number of container components
- **audio** (*integer*) – The number of audio components
- **video** (*integer*) – The number of video components
- **binary** (*integer*) – The number of binary components

**Accepts**

- **application/xml, application/json** – SimpleMetadataDocument

**Role** _import

### Shape files

### List all files for a shape

**GET /item/** (*id*) **/shape/**
*shape-id***/file** Returns all files that are associated with the specified shape.

**Query Parameters**

- **includeItem** (*boolean*) –
  - `true` - Return associated items, shapes, and components.
  - `false` (default) - Do not return any information about associated items, shapes, and components.
- **closedFiles** (*boolean*) –
  - `true` (default) - Return only files that are closed.
  - `false` - Return all files.
- **methodType** (*string*) – Return URIs from storage methods with a particular *type*. By default, return URLs with empty `methodType`.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodMetadata** (*string[]*) – *metadata* used with storage method.
- **scheme** (*string*) – URI scheme to return.

**Produces**

- **application/xml, application/json** – FileListDocument

**Role** _item_shape_read

### Shape metadata

Please refer to *Key-value metadata*.

### Updating an existing shape

If the shape deduction on import for some reason gave an incorrect result, it is possible to re-run the shape deduction using this command.

### Re-run a shape deduction on an existing shape

**POST /item/** (*item-id*) **/shape/**
*shape-id***/update** Starts a new shape deduction job for the specified shape.

**Query Parameters**

- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _item_shape_write

### Tags of a shape

### List all tags for a shape

**GET /item/** (*item-id*) **/shape/**
*shape-id***/tag/** Retrieves all shape tags associated with a certain shape.

**Query Parameters**

- **url** (*boolean*) –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – A list of the tags.

**Role** _shape_tag_read

### Add a tag to a shape

**PUT /item/** (*item-id*) **/shape/**
*shape-id***/tag/***tag-name* Adds shape tag with the given name to the specified shape. If the shape already has that tag, this operation does nothing.

**Status Codes**

- **200 OK** – Tag added successfully.
- **404 Not found** – No tag with that name exists.

**Role** _shape_tag_write

### Remove a tag from a shape

**DELETE /item/** (*item-id*) **/shape/**
*shape-id***/tag/***tag-name* Removes a tag with the given name from the specified shape.

**Status Codes**

- **200 OK** – Tag added successfully.
- **404 Not found** – No tag with that name exists within the shape.

**Role** _shape_tag_write

#### Shape mime types

#### List all mime types for a shape

**GET /item/** (*id*) **/shape/**
*shape-id***/mime/** Lists all mime types that are set on the shape. These can also be seen the ShapeDocument of the shape.

**Produces**

- **application/xml, application/json** – URIListDocument containing all the mime types of the shape.
- **text/plain** – CRLF-delimited list of mime types

**Role** _item_shape_read

#### Add a mime type to a shape

**PUT /item/** (*id*) **/shape/**
*shape-id***/mime/***mime-type* Adds a new mime type to the shape. This operation does nothing if the shape already has the mime-type.

**Role** _item_shape_write

#### Remove a mime type from a shape

**DELETE /item/** (*id*) **/shape/**
*shape-id***/mime/***mime-type* Removes a mime type from the shape.

**Role** _item_shape_write

### 17.11.8 Shape analysis

Shapes can be analyzed to detect for example detect cropping and silence.

Analysis will also generation information for generating waveform data, to be visualized in a UI.

> **Warning:** From version 5.4 and onward, sending an empty analyze request document will not perform an analysis of the shape. In earlier versions an empty document meant performing an analysis for all parameters, which could lead to excessive database disk usage.

#### Operations

#### Analyze a specific shape with job parameters in the request body

**POST /item/** (*item-id*) **/shape/**
*shape-id***/analyze** Analyzes the specified shape with the parameters specified in the job document. The result of the analyze will appear in the *bulky metadata* of the shape when doing a transcoder analysis, or in the *metadata* of the item when doing a cognitive service analysis.

**Query Parameters**

- **resourceId** (*string*) – The transcoder or cognitive service resource to use to execute the analysis.
- **storageId** (*string*) – The storage on which to store a temporary analysis data file when using a Vidinet transcoder to analyze a shape. If no storage id has been specified Vidispine

will (by default) automatically pick a supported storage. The storage id will be ignored when using a non Vidinet transcoder.

- **callbackId** (*string*) – The callback resource id to use for finding and running callback scripts.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Accepts**

- **application/xml, application/json** – AnalyzeJobDocument

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

### Example

Analyze a shape checking only for black frames:

```
POST /item/VX-123/shape/VX-456/analyze
Content-Type: application/xml

<AnalyzeJobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <black>
        <threshold>0.1</threshold>
    </black>
</AnalyzeJobDocument>
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-426</jobId>
  <user>admin</user>
  <started>2012-03-26T11:27:49.173Z</started>
  <status>READY</status>
  <type>ANALYZE</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

### Example

Analyze a shape with custom parameters:

```
POST /item/VX-124/shape/VX-457/analyze
Content-Type: application/xml

<AnalyzeJobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <black>
    <threshold>0.1</threshold>
    <percentage>95</percentage>
  </black>
  <freeze>
```

```
    <time>1.0</time>
    <threshold>0.05</threshold>
  </freeze>
  <bars>
    <percentage>10</percentage>
    <threshold>0.05</threshold>
  </bars>
</AnalyzeJobDocument>
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-427</jobId>
  <user>admin</user>
  <started>2012-03-26T11:27:49.173Z</started>
  <status>READY</status>
  <type>ANALYZE</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

In this example, settings for black frame, freeze and bar detection are included. The `threshold` elements determine the threshold to use when detecting black frames or freezes. The values have the following meaning:

- `threshold` for black frame detection and bar detection denotes that any pixel whose value is greater than `threshold` * 255 should not be regarded as black. I.e. only if `threshold` is 0 will only completely black pixels be counted.
- For freeze frame detection `threshold` determines how much any one pixel may change between two frames. If the difference in value between two frames is greater than `threshold` * 255, the frame will not be regarded as frozen.

### Example

Analyze a shape with audio parameters:

```
POST /item/VX-124/shape/VX-457/analyze
Content-Type: application/xml

<AnalyzeJobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <channel stream="1"/>
    <channel stream="2"/>
</AnalyzeJobDocument>
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-428</jobId>
  <user>admin</user>
  <started>2012-03-26T11:47:12.565Z</started>
  <status>READY</status>
  <type>ANALYZE</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

In this example, only the audio of the shape is analyzed. And specifically streams number 1 and 2. The numbers correspond to the *essenceStreamId* of the audio component. To analyze all streams, just add a single *channel* and omit the *stream* attribute.

### Analyze a specific shape using an analyze preset

**POST /item/**(*item-id*)**/shape/**

*shape-id*/**analyze** Analyzes the specified shape with the parameters from the analyze preset specified. The result of the analyze will appear in the *bulky metadata* of the shape when doing a transcoder analysis, or in the *metadata* of the item when doing a cognitive service analysis.

**Query Parameters**

- **preset** (*string*) – The analyze preset to use for the job
- **resourceId** (*string*) – The transcoder or cognitive service resource to use to execute the analysis.
- **storageId** (*string*) – The storage on which to store a temporary analysis data file when using a Vidinet transcoder to analyze a shape. If no storage id has been specified Vidispine will (by default) automatically pick a supported storage. The storage id will be ignored when using a non Vidinet transcoder.
- **callbackId** (*string*) – The callback resource id to use for finding and running callback scripts.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

### Analyze an item

**POST /item/** (*item-id*) **/analyze**

Analyzes an item with the parameters specified in the job document. The result of the analyze will appear in the *bulky metadata* of the shape when doing a transcoder analysis, or in the *metadata* of the item when doing a cognitive service analysis.

New in version 5.0.

**Query Parameters**

- **resourceId** (*string*) – The transcoder or cognitive service resource to use to execute the analysis.
- **tag** (*string*) – The shape tag to analyze. If omitted the original shape tag will be used.
- **storageId** (*string*) – The storage on which to store a temporary analysis data file when using a Vidinet transcoder to analyze a shape. If no storage id has been specified Vidispine will (by default) automatically pick a supported storage. The storage id will be ignored when using a non Vidinet transcoder.
- **callbackId** (*string*) – The callback resource id to use for finding and running callback scripts.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .

- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Accepts**

- **application/xml, application/json** – AnalyzeJobDocument

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

### Viewing the results

The results of the analysis is stored in the bulky metadata for the shape. For example, to view the black frame information (if available), go to `/item/VX-123/shape/VX-456/metadata/bulky/black`. You should see something like the following:

```
<BulkyMetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine" id="VX-295">
  <field stream="0" end="6@50" start="0@50">
    <key>black</key>
    <value>1</value>
  </field>
  <field stream="0" end="1650@50" start="1490@50">
    <key>black</key>
    <value>1</value>
  </field>
</BulkyMetadataDocument>
```

Each field contains a `start` and an `end` attribute, denoting the start and end timecodes for the black frames.

### Loudness analysis

When an analysis is done, a loudness analysis is done automatically. The result of the loudness analysis is written to the bulky metadata, but there are utility methods to easily extract the information.

### Get loudness values

**GET /item/** (*item-id*) **/loudness**

Extracts loudness information from bulky metadata.

**Produces**

- **application/xml, application/json** – LoudnessDocument

**Role** _item_shape_read

### Example

```
POST /item/VX-124/shape/VX-457/analyze
```

```
GET /item/VX-124/loudness
```

```
<LoudnessDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-124</id>
  <shape>VX-457</shape>
  <shapeTag>original</shapeTag>
  <mix>
    <name>Left</name>
```

```
    <weightdB>0.0</weightdB>
    <sourceStream>0</sourceStream>
    <sourceChannel>0</sourceChannel>
  </mix>
  <mix>
    <name>Right</name>
    <weightdB>0.0</weightdB>
    <sourceStream>0</sourceStream>
    <sourceChannel>1</sourceChannel>
  </mix>
  <mix>
    <name>Center</name>
    <weightdB>0.0</weightdB>
  </mix>
  <mix>
    <name>Left Surround</name>
    <weightdB>1.5</weightdB>
  </mix>
  <mix>
    <name>Right Surround</name>
    <weightdB>1.5</weightdB>
  </mix>
  <startLoudness>0@48000</startLoudness>
  <endLoudness>1339200@48000</endLoudness>
  <startRange>0@48000</startRange>
  <endRange>1296000@48000</endRange>
  <loudnessLU>0.014140396527686505</loudnessLU>
  <loudnessRangeLU>4.974758665644899</loudnessRangeLU>
</LoudnessDocument>
```

### Get loudness values for interval

**PUT /item/** (*item-id*) **/loudness**

Extracts loudness information from bulky metadata. Start and end range can be specified, as well as custom mixing.

**Accepts**

- **application/xml, application/json** – LoudnessDocument

**Produces**

- **application/xml, application/json** – LoudnessDocument

**Role** _item_shape_read

### Waveform information

The waveform data is not exactly the waveform, but measurements of the RMS values with a rather high sampling rate.

### Get waveform data

**GET /item/** (*item-id*) **/waveform/values**

Returns the waveform data as a WaveformDataDocument.

**Query Parameters**

- **itemTrack** (*string*) – The `itemTrack` value of the audio channel within the shape.

- **stream** (*string*) – The stream value of the audio channel within the component of the shape.
- **channel** (*string*) – The channel value of the audio channel within the stream of the component. If `itemTrack` and `stream` are omitted, this value can be used to denote tracks in a linear fashion, regardless of `itemTrack` and `stream`. Then `channel=0` means the first audio track, *channel=1'* the second, etc. Default is `0`.
- **shape** (*string*) – The shape id to use to get information from. If omitted the shape tag will be used. Note that an analysis of this shape must be done before the information is available.
- **tag** (*string*) – The shape tag to use. Default is `original`.
- **start** (*string*) – The start time code to get waveform information for. Default is `-INF`.
- **end** (*string*) – The end time code to get waveform information for. Default is `+INF`.
- **dB** (*boolean*) –
  - `true` - Return RMS dB values.
  - `false` (default) - Return RMS 1-based absolute values.
- **width** (*integer*) – The number of sample points to return. Default is `400`.

**Produces**

- **application/xml, application/json** – WaveformDataDocument

**Role** _item_shape_read

### Get waveform data (deprecated)

Deprecated since version 4.5.1.

**GET /item/** (*item-id*) **/waveform/data**
Returns the waveform data as a JSON array.

**Query Parameters**

- **itemTrack** (*string*) – The `itemTrack` value of the audio channel within the shape.
- **stream** (*string*) – The stream value of the audio channel within the component of the shape.
- **channel** (*string*) – The channel value of the audio channel within the stream of the component. If `itemTrack` and `stream` are omitted, this value can be used to denote tracks in a linear fashion, regardless of `itemTrack` and `stream`. Then `channel=0` means the first audio track, *channel=1'* the second, etc. Default is `0`.
- **shape** (*string*) – The shape id to use to get information from. If omitted the shape tag will be used. Note that an analysis of this shape must be done before the information is available.
- **tag** (*string*) – The shape tag to use. Default is `original`.
- **start** (*string*) – The start time code to get waveform information for. Default is `-INF`.
- **end** (*string*) – The end time code to get waveform information for. Default is `+INF`.
- **dB** (*boolean*) –
  - `true` - Return RMS dB values.
  - `false` (default) - Return RMS 1-based absolute values.

- **width** (*integer*) – The number of sample points to return. Default is `400`.

**Produces**

- **application/json** – A JSON array with one JSON object. The JSON object contains one value with key `data`. The value is a JSON array with `width` number of data points.

**Role** _item_shape_read

### Get waveform image

**GET /item/**(*item-id*)**/waveform/image**

Returns an image with the waveform drawn on the canvas as described by the query parameters.

**Query Parameters**

- **itemTrack** (*string*) – The `itemTrack` value of the audio channel within the shape.
- **stream** (*string*) – The stream value of the audio channel within the component of the shape.
- **channel** (*string*) – The channel value of the audio channel within the stream of the component. If `itemTrack` and `stream` are omitted, this value can be used to denote tracks in a linear fashion, regardless of `itemTrack` and `stream`. Then `channel=0` means the first audio track, *channel=1* the second, etc. Default is `0`.
- **shape** (*string*) – The shape id to use to get information from. If omitted the shape tag will be used. Note that an analysis of this shape must be done before the information is available.
- **tag** (*string*) – The shape tag to use. Default is `original`.
- **start** (*string*) – The start time code to get waveform information for. Default is `-INF`.
- **end** (*string*) – The end time code to get waveform information for. Default is `+INF`.
- **dB** (*boolean*) –
  - `true` - Return RMS dB values.
  - `false` (default) - Return RMS 1-based absolute values.
- **width** (*integer*) – The number of sample points to return. Default is `400`.
- **height** (*integer*) – The height, in pixels, of the image. Default is 100.
- **bgcolor** (*string*) – The background color of the image, as hex triplet. Default is `#000000` (black).
- **fgcolor** (*string*) – The color of the waveform, as hex triplet. Default is `#ffffff` (white).
- **hgridline** (*string*) – The position of primary horizontal gridlines, in units of the audio. Default is `""` (no gridline).
- **hgridlinecolor** (*string*) – The color of primary horizontal gridlines. Default is `#808080`.
- **hgridline2** (*string*) – The position of secondary horizontal gridlines, in units of the audio. Default is `""` (no gridline).
- **hgridline2color** (*string*) – The color of secondary horizontal gridlines. Default is `#404040`

- **vgridline** (*string*) – The position of primary vertical gridlines, where 0 is left border and 1 is right border. Default is `""` (no gridline).
- **vgridlinecolor** (*string*) – The color of primary vertical gridlines. Default is `#808080`
- **vgridline2** (*string*) – The position of primary vertical gridlines, where 0 is left border and 1 is right border. Default is `""` (no gridline).
- **vgridline2color** (*string*) – The color of primary vertical gridlines. Default is `#404040`
- **min** (*number*) – The audio value that corresponds the bottom border. Defaults to `-1` if `dB` is `false`, and `-80` otherwise.
- **max** (*number*) – The audio value that corresponds the top border. Defaults to `1` if `dB` is `false`, and `0` otherwise.

**Produces**

- **image/png** – A PNG image.

**Role** _item_shape_read

### Get waveform image URI

**GET /item/** (*item-id*) **/waveform/imageURI**

Returns a URI that does not require authentication to the generated image. The URI expires after 1 hour.

**Query Parameters**

- **itemTrack** (*string*) – The `itemTrack` value of the audio channel within the shape.
- **stream** (*string*) – The stream value of the audio channel within the component of the shape.
- **channel** (*string*) – The channel value of the audio channel within the stream of the component. If `itemTrack` and `stream` are omitted, this value can be used to denote tracks in a linear fashion, regardless of `itemTrack` and `stream`. Then `channel=0` means the first audio track, *channel=1'* the second, etc. Default is `0`.
- **shape** (*string*) – The shape id to use to get information from. If omitted the shape tag will be used. Note that an analysis of this shape must be done before the information is available.
- **tag** (*string*) – The shape tag to use. Default is `original`.
- **start** (*string*) – The start time code to get waveform information for. Default is `-INF`.
- **end** (*string*) – The end time code to get waveform information for. Default is `+INF`.
- **dB** (*boolean*) –
  - `true` - Return RMS dB values.
  - `false` (default) - Return RMS 1-based absolute values.
- **width** (*integer*) – The number of sample points to return. Default is `400`.
- **height** (*integer*) – The height, in pixels, of the image. Default is 100.
- **bgcolor** (*string*) – The background color of the image, as hex triplet. Default is `#000000` (black).

- **fgcolor** (*string*) – The color of the waveform, as hex triplet. Default is `#ffffff` (white).
- **hgridline** (*string*) – The position of primary horizontal gridlines, in units of the audio. Default is `""` (no gridline).
- **hgridlinecolor** (*string*) – The color of primary horizontal gridlines. Default is `#808080`.
- **hgridline2** (*string*) – The position of secondary horizontal gridlines, in units of the audio. Default is `""` (no gridline).
- **hgridline2color** (*string*) – The color of secondary horizontal gridlines. Default is `#404040`
- **vgridline** (*string*) – The position of primary vertical gridlines, where 0 is left border and 1 is right border. Default is `""` (no gridline).
- **vgridlinecolor** (*string*) – The color of primary vertical gridlines. Default is `#808080`
- **vgridline2** (*string*) – The position of primary vertical gridlines, where 0 is left border and 1 is right border. Default is `""` (no gridline).
- **vgridline2color** (*string*) – The color of primary vertical gridlines. Default is `#404040`
- **min** (*number*) – The audio value that corresponds the bottom border. Defaults to `-1` if `dB` is `false`, and `-80` otherwise.
- **max** (*number*) – The audio value that corresponds the top border. Defaults to `1` if `dB` is `false`, and `0` otherwise.

**Produces**

- **text/plain** – The URI to the image.

**Role** _item_shape_read

### Get waveform images for all audio channels

**GET /item/** (*item-id*) **/waveform/alltracks**

Solely used for debugging. May be deprecated in newer releases.

Returns a HTML document including image references to waveform images for all channels. Query parameters can be used to control the image appearance.

**Query Parameters**

- **itemTrack** (*string*) – The `itemTrack` value of the audio channel within the shape.
- **stream** (*string*) – The stream value of the audio channel within the component of the shape.
- **channel** (*string*) – The channel value of the audio channel within the stream of the component. If `itemTrack` and `stream` are omitted, this value can be used to denote tracks in a linear fashion, regardless of `itemTrack` and `stream`. Then `channel=0` means the first audio track, *channel=1'* the second, etc. Default is `0`.
- **shape** (*string*) – The shape id to use to get information from. If omitted the shape tag will be used. Note that an analysis of this shape must be done before the information is available.
- **tag** (*string*) – The shape tag to use. Default is `original`.

- **start** (*string*) – The start time code to get waveform information for. Default is `-INF`.
- **end** (*string*) – The end time code to get waveform information for. Default is `+INF`.
- **dB** (*boolean*) –
  - `true` - Return RMS dB values.
  - `false` (default) - Return RMS 1-based absolute values.
- **width** (*integer*) – The number of sample points to return. Default is `400`.
- **height** (*integer*) – The height, in pixels, of the image. Default is 100.
- **bgcolor** (*string*) – The background color of the image, as hex triplet. Default is `#000000` (black).
- **fgcolor** (*string*) – The color of the waveform, as hex triplet. Default is `#ffffff` (white).
- **hgridline** (*string*) – The position of primary horizontal gridlines, in units of the audio. Default is `""` (no gridline).
- **hgridlinecolor** (*string*) – The color of primary horizontal gridlines. Default is `#808080`.
- **hgridline2** (*string*) – The position of secondary horizontal gridlines, in units of the audio. Default is `""` (no gridline).
- **hgridline2color** (*string*) – The color of secondary horizontal gridlines. Default is `#404040`
- **vgridline** (*string*) – The position of primary vertical gridlines, where 0 is left border and 1 is right border. Default is `""` (no gridline).
- **vgridlinecolor** (*string*) – The color of primary vertical gridlines. Default is `#808080`
- **vgridline2** (*string*) – The position of primary vertical gridlines, where 0 is left border and 1 is right border. Default is `""` (no gridline).
- **vgridline2color** (*string*) – The color of primary vertical gridlines. Default is `#404040`
- **min** (*number*) – The audio value that corresponds the bottom border. Defaults to `-1` if `dB` is `false`, and `-80` otherwise.
- **max** (*number*) – The audio value that corresponds the top border. Defaults to `1` if `dB` is `false`, and `0` otherwise.

**Produces**

- **text/html** – A HTML document.

**Role** _item_shape_read

### Highlights extraction

New in version 5.4.

Utilizing Nablet Shrynk, an analysis job can calculate an "interest factor" for each frame in a video. This data can then be used to create a highlight reel. There are a number of AI models built in, and in order to get good results it is important to use a model that is tailored to your content. Currently, these models are available:

- `football`

- ice-hockey
- basketball
- formula1
- handball

### Example

Analyze a shape with default parameters:

```
POST /item/VX-123/shape/VX-456/analyze
Content-Type: application/xml

<AnalyzeJobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <highlighter>
    <model>football</model>
  </highlighter>
</AnalyzeJobDocument>
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-429</jobId>
  <user>admin</user>
  <started>20120-08-20T11:27:49.173Z</started>
  <status>READY</status>
  <type>ANALYZE</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

### Get a highlight reel EDL

**GET /item/** (*item-id*) **/shape/**
*shape-id***/highlighter-edl** If a highligheter analysis has been performed, this returns a conform document for producing a highlight reel of the desired duration.

**Query Parameters**

- **duration** (*integer*) – The desired duration of the highlight reel in seconds.

**Produces**

- **application/xml, application/json** – ConformDocument

**Role** _item_read

### Start a highlight reel creation job

**POST /item/** (*item-id*) **/shape/**
*shape-id***/highlight-render** Starts a highlight render job for the given shape, producing an output file of the desired duration.

**Query Parameters**

- **duration** (*integer*) – The desired duration of the highlight reel in seconds.
- **conformMetadata** (*boolean*) –
  - `true` (default) - Copy metadata from the source items, according to the timeline, to the resulting item.

  - `false` - Do not copy metadata from the source items.
- **sourceTag** (*string*) – Comma-separated list of shape tags, that specify the shapes that should be used as inputs.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **tag** (*string*) – Comma-separated list of shape tags, that specify the desired output.
- **createThumbnails** (*boolean*) –
  - `true` (default) - Creates thumbnails according to default transcoder rules.
  - `false` - No thumbnails will be created.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **destinationItem** (*string*) – An item id, to which the new new shape will be associated.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM`.
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

### Smart cropping

New in version 5.4.

Utilizing Nablet Heightscreen, landscape format content can be cropped into portrait mode, using an AI algorithm that automatically determines the areas of highest interest in the video.

### Example

Analyze a shape with default parameters:

```
POST /item/VX-123/shape/VX-456/analyze
Content-Type: application/xml

<AnalyzeJobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <smartcrop>
    <aspect>9:16</aspect>
  </smartcrop>
</AnalyzeJobDocument>
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-430</jobId>
  <user>admin</user>
```

```
  <started>20120-08-20T12:12:42.758Z</started>
  <status>READY</status>
  <type>ANALYZE</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

### Get a smart cropping EDL

**GET /item/** (*item-id*) **/shape/**
*shape-id***/smartcrop-edl** If a smartcrop analysis has been performed, this returns a sequence document for producing an output with the desired aspect ratio.

**Query Parameters**

- **aspect** (*string*) – The aspect ratio to use, substituting "_" for ":" (e.g. `9_16`). An analyze job for this aspect ratio must have been run prior to making this request.

**Produces**

- **application/xml, application/json** – SequenceDocument

**Role** _item_read

### Start a smart cropping job

**POST /item/** (*item-id*) **/shape/**
*shape-id***/smartcrop-render** Starts a smart cropping render job, producing an output with the desired aspect ratio.

**Query Parameters**

- **aspect** (*string*) – The aspect ratio to use, substituting "_" for ":" (e.g. `9_16`). An analyze job for this aspect ratio must have been run prior to making this request.
- **tag** (*string[]*) – The shape tag specifying the format of the rendered sequence.
- **sourceTag** (*string[]*) – The shape tag specifying the shapes to use as input.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **destinationItem** (*string*) – An item id, to which the new new shape will be associated.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.

  New in version 4.16.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

### 17.11.9 Shape components

#### Components

#### List all components for a shape

**GET /item/** (*id*) **/shape/**
*shape-id***/component** Returns all components for a specified shape. Currently, this call returns the same information as the return shape, but is available for orthogonality.

**Status Codes**

- **404 Not found** – Invalid id

**Produces**

- **application/xml, application/json** – ComponentListDocument

**Role** _item_shape_read

#### Retrieve a component

**GET /item/** (*id*) **/shape/**
*shape-id***/component/***component-id* Returns all files, or the complete component information, for a specified component.

**Query Parameters**

- **full** (*boolean*) –
  - `true` - Return the component information.
  - `false` (default) - Return all files.

**Produces**

- **application/xml, application/json** – ComponentDocument if `full=true`, else a *FileListDocument*.
- **text/plain** – List of file URLs

**Role** _item_shape_read

#### Component import

#### Import a component using a URI or an existing file

**POST /item/** (*id*) **/shape/**
*shape-id***/component** Starts a job that imports a component to an existing shape. The shape must be a media shape and must not be a placeholder.

**Query Parameters**

- **uri** (*string*) – The URI to the file containing the new shape.
- **fileId** (*string*) – The id of the file that contains the new shape.
- **allowReimport** (*boolean*) –
  - `true` - Import the file to this shape even if the file is already importing or is already part of another item.
  - `false` (default) Reject the request if the file with the given id has already been imported or is currently importing.

- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Status Codes**

- **400** – If the file has already been imported.

**Role** _import

### Move/copy components

### Move a component to another shape

**POST /item/**(*id*)**/shape/***shape-id***/component/***component-id***/move/item/***target-id***/shape/***target-shape-id* Move this component to another shape.

**Query Parameters**

- **index** (*integer*) – The component index (track) of component. If the target shape has a component with this index, then it will be replaced/removed.
- **keepMetadata** (*boolean*) –
  - `true` - Preserve the metadata from the replaced component.
  - `false` (default) - Discard any metadata from the replaced component.

**Produces**

- **application/xml, application/json** – ShapeDocument from the target shape.

**Role** _item_shape_write

### Move a component to another shape/component

**POST /item/**(*id*)**/shape/***shape-id***/component/***component-id***/move/item/***target-id***/shape/***target-shape-id***/component/***target-component-id* Move this component to another shape, replacing a specific component by id.

**Query Parameters**

- **index** (*integer*) – The component index (track) of component. If the target shape has a component with this index, then it will be replaced/removed.
- **keepMetadata** (*boolean*) –
  - `true` - Preserve the metadata from the replaced component.
  - `false` (default) - Discard any metadata from the replaced component.

**Produces**

- **application/xml, application/json** – ShapeDocument from the target shape.

**Role** _item_shape_write

### Copy a component to another shape

**POST /item/**(*id*)**/shape/***shape-id***/component/***component-id***/copy/item/***target-id***/shape/***target-shape-id* Copy this component to another shape.

**Query Parameters**

- **index** (*integer*) – The component index (track) of component. If the target shape has a component with this index, then it will be replaced/removed.
- **keepMetadata** (*boolean*) –
  - `true` - Preserve the metadata from the replaced component.
  - `false` (default) - Discard any metadata from the replaced component.

**Produces**

- **application/xml, application/json** – ShapeDocument from the target shape.

**Role** _item_shape_write

### Copy a component to another shape/component

**POST /item/**(*id*)**/shape/***shape-id***/component/***component-id***/copy/item/***target-id***/shape/***target-shape-id***/component/***target-component-id* Copy this component to another shape, replacing a specific component by id.

**Query Parameters**

- **index** (*integer*) – The component index (track) of component. If the target shape has a component with this index, then it will be replaced/removed.
- **keepMetadata** (*boolean*) –
  - `true` - Preserve the metadata from the replaced component.
  - `false` (default) - Discard any metadata from the replaced component.

**Produces**

- **application/xml, application/json** – ShapeDocument from the target shape.

**Role** _item_shape_write

### Delete a component

**DELETE /item/**(*id*)**/shape/***shape-id***/component/***component-id* Removes the component from the shape. Any files belonging to the component is not Copy this component to another shape, replacing a specific component by id.

**Query Parameters**

- **keepFiles** (*boolean*) –
  - `true` - Keep the files belong to this shape.
  - `false` (default) - Remove the files belong to this shape.

**Role** _item_shape_write

#### Component files

#### Associate a file with a component

**PUT /item/** (*id*) **/shape/**
*shape-id***/component/***component-id***/file/***file-id** Attaches the specified file to the specified component

**Query Parameters**

- **allowReimport** (*boolean*) –
  - `true` - Associate the file regardless of whether it already belongs to a component.
  - `false` (default) - Only files that do not already belong to a component can be associated.

**Produces**

- **application/xml, application/json** – ComponentDocument

**Role** _item_shape_write

#### Remove a file from a component

**DELETE /item/** (*id*) **/shape/**
*shape-id***/component/***component-id***/file/***file-id** Removes the specified file from the specified component

**Role** _item_shape_write

#### Placeholder components

#### Create a placeholder component

**POST /item/** (*id*) **/shape/**
*shape-id***/component/placeholder** Creates an new placeholder component for a specific shape.

**Query Parameters**

- **type** (*string*) – The type of component. Required. One of `audio`, `video`, `container` or `binary`.
- **index** (*integer*) – The component index (track) of new component.

**Produces**

- **application/xml, application/json** – ComponentDocument

#### Component metadata

Please refer to *Key-value metadata*.

### 17.11.10 Thumbnails

#### Creating thumbnails and posters

Thumbnails and posters can be created by starting a thumbnail job.

#### Create a thumbnail job

**POST /item/** (*item-id*) **/thumbnail**
Creates a new thumbnail job with the specified parameters. Note that a job cannot both create thumbnails at specified intervals and posters. Creating thumbnails according to transcoder rules and creating posters is however allowed.

Changed in version 5.0: For multi-layer PSD/PSB files, only a thumbnail of all layers flattened will be generated by default.

**Query Parameters**

- **createThumbnails** (*boolean*) –
  - `true` - Creates thumbnails according to default transcoder rules.
  - *t1*, ... - Thumbnails will be created on the specified, comma-separated, *time codes*.
  - `false` (default) - No thumbnails will be created.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **thumbnailWidth** (*integer*) – The width of the thumbnails. If `thumbnailWidth` is specified, `thumbnailHeight` must also be specified.
- **thumbnailHeight** (*integer*) – The height of the thumbnails. If `thumbnailHeight` is specified, `thumbnailWidth` must also be specified.
- **thumbnailPeriod** (*string*) – Timecode string specifying the interval of the thumbnails. It should be a decimal integer when working with multi-page images/PDFs, meaning every N page(s).
- **posterWidth** (*integer*) – The width of the posters.
- **posterHeight** (*integer*) – The height of the posters.
- **posterFormat** (*string*) –
  - `jpeg` (default) - Creates posters in JPEG format.
  - `png` - Creates posters in PNG format.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **tag** (*string*) – Include additional video settings from this transcode preset. Resolution settings in the tag are overridden by query parameters `thumbnailHeight` and `thumbnailWidth`.
- **version** (*integer*) – A version number. For creating thumbnails for older versions of the item essence. Default is latest version.
- **sourceTag** (*string*) – Comma-separated shape tags. The fist valid shape will be chosen as the source of the job. If non of the tags are valid, the original shape will be used.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

### Example

Creating thumbnails according to transcoder rules and posters at the time codes 50@PAL and 100@PAL.

```
POST /item/VX-123/thumbnail?createThumbnails=true&createPosters=50@PAL,100@PAL&
↪sourceTag=mov,mp4
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <jobId>VX-1219</jobId>
   <user>admin</user>
   <started>2010-04-23T11:24:24.434+02:00</started>
   <status>READY</status>
   <type>THUMBNAIL</type>
   <priority>MEDIUM</priority>
</JobDocument>
```

### Item thumbnail resources

The following requests deal with managing thumbnail resources for specific items.

### List thumbnail resources for an item

**GET /item/** (*item-id*) **/thumbnailresource**
Return one or more poster resource URIs which can be used to manage the thumbnails for a specific item.

**Query Parameters**

- **version** (*integer*) – Return thumbnails from this essence version. By default thumbnails for the latest version will be returned.

**Produces**

- **text/plain** – CRLF-delimited list of thumbnail resource URIs
- **application/xml, application/json** – URIListDocument of thumbnail resource URIs

**GET /item/** (*item-id*) **/posterresource**
Return one or more poster resource URIs which can be used to manage the posters for a specific item.

**Query Parameters**

- **version** (*integer*) – Return posters from this essence version. By default posters for the latest version will be returned.

**Produces**

- **text/plain** – CRLF-delimited list of thumbnail resource URIs
- **application/xml, application/json** – URIListDocument of thumbnail resource URIs

### Update or create a thumbnail resource for an item

**PUT /item/** (*item-id*) **/thumbnailresource**
If no thumbnail resources are defined for an item, create a resource and return it.

**Produces**

- **text/plain** – CRLF-delimited list of thumbnail resource URIs

- **application/xml, application/json** – URIListDocument of thumbnail resource URIs

**PUT /item/** (*item-id*) **/posterresource**
If no poster resources are defined for an item, create a resource and return it.

**Produces**

- **text/plain** – CRLF-delimited list of thumbnail resource URIs
- **application/xml, application/json** – URIListDocument of thumbnail resource URIs

---

**Note:** Thumbnails and posters for an item share the same resource. Hence, if a resource is added for posters, it is automatically added for thumbnails as well.

---

### Get a thumbnail sprite sheet for an item

New in version 5.6.

A thumbnail sprite sheet is a large image containing all of the thumbnails for an item, together with information about each thumbnail.

**GET /item/** (*item-id*) **/thumbnail/spritesheet**
Returns a thumbnail sprite sheet which contains a URI to the generated sprite sheet and the positions of the images.

**Query Parameters**

- **noauth-url** (*boolean*) –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs

**Request Headers**

- **If-Modified-None** – Optional header containing ETag of previous call. If no changes to thumbnails have been done, a 304 Not Modified is returned.

**Response Headers**

- **ETag** – Contains the computed ETag of the response.

**Produces**

- **application/xml** – ThumbnailSpriteSheetDocument
- **text/vtt** – The sheet in WebVTT form

### Thumbnail resource handling

The following requests deal with managing collections of thumbnail URIs for a specific thumbnail resource.

### List all thumbnails

**GET {thumbnail-resource}**
Returns thumbnail URIs on which further requests may be performed.

**Query Parameters**

- **url** (*boolean*) –

- `true` - Return list of URLs.
- `false` (default) - Return list of ids.

- **noauth-url** (*boolean*) –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs

**Produces**

- **text/plain** – CRLF-delimited list of thumbnail URIs.
- **application/xml, application/json** – URIListDocument of thumbnail URIs

**Role** _thumbnail_read

### Update or create a thumbnail

**PUT {thumbnail-resource}/**(*time*)
Create a new thumbnail at the specified time code. If a thumbnail with the specified time code already exists it is replaced.

**Accepts**

- **image/png, image/jpeg** – Image to insert

**Produces**

- **text/plain** – Informational status message.

**Status Codes**

- **400** – Given data was not valid `image/png` or `image/jpeg`

**Role** _thumbnail_write

### Delete all thumbnails

**DELETE {thumbnail-resource}**
Remove all thumbnails handled by this resource.

**Role** _thumbnail_write

### Thumbnail handling

The following requests concern handling a specific thumbnail.

### Retrieve the image representation

**GET {thumbnail-resource}/**(*time*)
Return the image representation of this thumbnail.

**Query Parameters**

- **hash** (*string*) – The checksum of the image.
- **type** (*string*) – Optional type.

**Produces**

- **image/png, image/jpeg** – Image of the thumbnail

**Role** _thumbnail_read

#### Delete a thumbnail

**DELETE {thumbnail-resource}/**(*time*)
Remove this thumbnail.

**Role** _thumbnail_write

#### Export a thumbnail

**POST {thumbnail-resource}/**(*time*)**/export**
Starts a job that writes the thumbnail or poster to a specific destination.

**Query Parameters**

- **uri** (*string*) – Required. URI of export location of thumbnail or poster
- **format** (*string*) – Image format of destination. E.g. `tiff`.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _thumbnail_read

### 17.11.11 Transcoding

#### Transcoding

#### Transcode an item

**POST /item/**(*item-id*)**/transcode**
Starts a new job that transcode an item to a number of shapes according to the given shape tags.

**Query Parameters**

- **tag** (*string*) – Comma-separated list of shape tags, that specify the desired output.
- **createThumbnails** (*boolean*) –
  - `true` (default) - Creates thumbnails according to default transcoder rules.
  - `false` - No thumbnails will be created.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **destinationItem** (*string*) – An item id, to which the new new shape will be associated.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.

- **storageId** (*string*) – Identifier of storage where essence file is to be stored.

  New in version 4.16.

- **overrideFastStart** (*boolean*) –
  - `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
  - `false` - Do not use the transcoder's estimate of the duration.
- **requireFastStart** (*boolean*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** (*string*) – Estimated duration of the clip in seconds.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

### Transcode a specific shape

**POST /item/** (*item-id*) **/shape/** *shape-id***/transcode** Starts a new job that transcode a specific shape on an item to a number of shapes according to the given shape tags.

**Query Parameters**

- **tag** (*string*) – Comma-separated list of shape tags, that specify the desired output.
- **createThumbnails** (*boolean*) –
  - `true` (default) - Creates thumbnails according to default transcoder rules.
  - `false` - No thumbnails will be created.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **destinationItem** (*string*) – An item id, to which the new new shape will be associated.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **storageId** (*string*) – Identifier of storage where essence file is to be stored.

  New in version 4.16.

- **overrideFastStart** (*boolean*) –

- `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
- `false` - Do not use the transcoder's estimate of the duration.

- **requireFastStart** (*boolean*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** (*string*) – Estimated duration of the clip in seconds.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

### 17.11.12 Item conform

The conform resource exposes a simple way to combine media from one or more items into a new item. It is also possible to select specific parts from the input by specifying an input interval.

This could be used when you for example want to:

- Merge spanned P2 clips.
- Render a simple sequence as defined by a user.

#### Conforming

#### Start a conform job

**POST /conform**

Starts a new `CONFORM` job that creates a new item and one or more shapes that contains media according to the conform timeline.

**Query Parameters**

- **conformMetadata** (*boolean*) –
  - `true` (default) - Copy metadata from the source items, according to the timeline, to the resulting item.
  - `false` - Do not copy metadata from the source items.
- **sourceTag** (*string*) – Comma-separated list of shape tags, that specify the shapes that should be used as inputs.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **tag** (*string*) – Comma-separated list of shape tags, that specify the desired output.
- **createThumbnails** (*boolean*) –

- `true` (default) - Creates thumbnails according to default transcoder rules.
- `false` - No thumbnails will be created.

- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **destinationItem** (*string*) – An item id, to which the new new shape will be associated.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Accepts**

- **application/xml, application/json** – ConformRequestDocument

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

### Start a conform job for an existing item

**POST /item/** (*id*) **/timeline/** *timeline-format* **/conform** Starts a new `CONFORM` job that creates one or more shapes that contains media according to the conform timeline.

The timeline must be a ConformDocument, else the request will be rejected.

**Query Parameters**

- **conformMetadata** (*boolean*) –
  - `true` (default) - Copy metadata from the source items, according to the timeline, to the resulting item.
  - `false` - Do not copy metadata from the source items.
- **sourceTag** (*string*) – Comma-separated list of shape tags, that specify the shapes that should be used as inputs.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **tag** (*string*) – Comma-separated list of shape tags, that specify the desired output.
- **createThumbnails** (*boolean*) –
  - `true` (default) - Creates thumbnails according to default transcoder rules.
  - `false` - No thumbnails will be created.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.

- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **destinationItem** (*string*) – An item id, to which the new new shape will be associated.
- **overrideFastStart** (*boolean*) –
  - `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
  - `false` - Do not use the transcoder's estimate of the duration.
- **requireFastStart** (*boolean*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** (*string*) – Estimated duration of the clip in seconds.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

### Example: joining two clips

In this example item `VX-1` and `VX-2` are the two clips that should be joined/concatenated. A shape-tag with an empty preset (here the `original` tag) is specified so that the output has the same format as the input.

```
POST /conform?tag=original
Content-Type: application/xml

<ConformRequestDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <conform>
    <timeline>
      <segment>
        <source>
          <id>VX-1</id>
        </source>
      </segment>
      <segment>
        <source>
          <id>VX-2</id>
        </source>
      </segment>
    </timeline>
  </conform>
  <metadata>
    <timespan start="-INF" end="+INF">
      <field>
        <name>title</name>
```

```
        <value>Joined A and B</value>
      </field>
    </timespan>
  </metadata>
</ConformRequestDocument>
```

```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-21527</jobId>
  <user>admin</user>
  <started>2013-05-14T06:56:40.868Z</started>
  <status>READY</status>
  <type>CONFORM</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

### 17.11.13 Timeline

Timeline API is used to store and retrieve timelines in the system. A timeline is represented in the system as a combination of:

- The timeline itself (in the native format, but can be converted to other formats)
- The rendered result as a new item.
- The timeline with its constituent parts as a collection.

There is no special datatype for the timeline, instead it is stored in one or several metadata fields.

#### Get Information about Item Timeline

#### List all timelines

**GET /item/**(*id*)**/timeline**
Returns a list of timeline stored and all timeline formats that can be derived. The precision number returned is an estimation of how close the converted timeline format will match the original. The lowest number, 0, means the original timeline format, 1 means derived timeline without any loss of information compared to the original format. The highest number, 100, means straight cuts only.

**Produces**

- **text/plain** – CRLF-delimited list of TabbedTuple of timeline format and precision.

**Role** _item_timeline_read

**See also:**

**CRLF** is used in `text/plain` representation when several values are returned, such as tuples or lists. CRLF is represented by the two bytes 0d 0a in hexadecimal notation.

**See also:**

**TabbedTuple** is used in `text/plain` representation when several values are returned, such as tuples or lists. TabbedTuples delimits each value by the tab character, `09` in hexadecimal notation. Together with `CRLF` it is used to create lists of tuples. Users should ignore any output after the last defined element in the tuple, more elements may be returned in future versions of the API.

#### Retrieve a timeline

**GET /item/**(*id*)**/timeline/**
*timeline-format* Returns timeline.

**Produces**

- `*/*` – Timeline representation. The actual MIME type depends on timeline format.

**Status Codes**

- `400` – Could not find the timeline identified by timeline-format.

**Role** _item_timeline_read

#### Update or create a timeline

**`PUT /item/`** (*id*) **`/timeline/`**
*timeline-format* updates or creates a timeline representation for an item.

**Accepts**

- `*/*` – Timeline representation. The actual MIME type depends on timeline format.

**Status Codes**

- `200` – The timeline was created/modified successfully.

**Role** _item_timeline_write

#### Delete all timelines

**`DELETE /item/`** (*id*) **`/timeline`**
Removes all timeline representations.

**Status Codes**

- `200` – All timelines associated with the item were deleted successfully.

**Role** _item_timeline_write

#### Delete a timeline

**`DELETE /item/`** (*id*) **`/timeline/`**
*timeline-format* Removes specific timeline representation.

**Status Codes**

- `200` – The timeline was deleted successfully.
- `400` – Could not find the timeline identified by timeline-format.

**Role** _item_timeline_write

## 17.12 JavaScript

Test and debug JavaScript.

### 17.12.1 Testing scripts

In order to test functionality, the JavaScript engine can be called manually.

#### Execute JavaScript

**POST /javascript/test**
Executes the given JavaScript code.

**Accepts**

- **application/javascript , text/plain, text/javascript** – The JavaScript code

**Produces**

- **application/json** – The object returned by the code, in JSON format

**Role** _administrator

### 17.12.2 JavaScript sessions

See *Debugging JavaScript* on how to debug JavaScript scripts.

#### List all JavaScript sessions

**GET /javascript/session**
Retrieves the the current JavaScript sessions, their current status, and which port they listen to.

**Produces**

- **text/plain** – A textual list of the current JavaScript sessions in Vidispine, their current status, and which port they listen to

**Role** _administrator

---

**Note:** Multiple JavaScript debug sessions cannot share the same port. If several JavaScript sessions are started simultaneously, each is allocate the next free port. The port number can be found using the request above.

---

#### Example

```
POST /javascript/test
Content-Type: application/javascript

1+1
```

```
GET /javascript/session
```

```
0    testscript      STARTED/RUNNING 0.0.0.0/0.0.0.0:59000   null
```

#### Retrieve a JavaScript session

**GET /javascript/session/** (*id*)
Retrieves stack trace of a specific JavaScript session.

**Produces**

- **text/plain** – The stack trace of the session

**Status Codes**

- **400** – The session has no frames

- **404** – No such session

**Role** _administrator

#### Example

After having connected to the session:

```
GET /javascript/session
```

```
0    testscript      CONNECTED/SUSPENDED     0.0.0.0/0.0.0.0:59000   testscript-
↪213efccdd3.js:2
```

```
GET /javascript/session/0
```

```
testscript-213efccdd3.js:2
```

#### Stop a JavaScript session

**DELETE /javascript/session/** (*id*)

Stops an active debugging session.

**Query Parameters**

- **stop** (*boolean*) –
  - `true` - Kill the running script.
  - `false` (default) - Stop the debugging session, and leave the script running.

**Status Codes**

- **400** – If no such session could be found.

**Role** _administrator

## 17.13 Jobs

Jobs make up the long running tasks in Vidispine. They are created in response to requests that would otherwise not be able to respond in time, such as import, export and transcode requests.

### 17.13.1 Managing jobs

#### Create a job

See *Creating jobs*.

#### List all jobs

**GET /job**

Return jobs matching the criteria given.

Changed in version 5.1: The priority parameter was added.

**Query Parameters**

- **jobmetadata** (*string[]*) – Multiple query parameters can be specified. If no query parameters are specified, all jobs are returned.
  - *key* = *value* - Filter out only the jobs that has job metadata according to the filter criteria.

Note: the metadata field `item` is generated and cannot be filtered on, instead the field `itemId` should be used.

Note: = is part of the query parameter, and has to be encoded (`%3d`).

- **metadata** (*boolean*) –
  - `true` - Include job metadata with all jobs.
  - `false` (default) - Do not include job metadata with all jobs.
- **idonly** (*boolean*) –
  - `true` - Only return a list of ids
  - `false` (default) - Return job information such as job status
- **starttime-from** (*string*) – ISO 8601 timestamp. Return only jobs started after and at the given timestamp.
- **starttime-to** (*string*) – ISO 8601 timestamp. Return only jobs started before and at the given timestamp.
- **finishtime-from** (*string*) – ISO 8601 timestamp. Return only jobs finished after and at the given timestamp.
- **finishtime-to** (*string*) – ISO 8601 timestamp. Return only jobs finished before and at the given timestamp.
- **step** (*boolean*) –
  - `true` - Include step information in the job listing.
  - `false` (default) - Do not include step information in the job listing.
- **type** (*string*) – Comma-separated list of *job types*. Default is `all`, return all jobs.
- **state** (*string*) – Comma-separated list of *job states*. Default is `all`, return all jobs.
- **priority** (*string*) – Comma-separated list of *job priorities*. Default is `all`, return all jobs.
- **first** (*integer*) – Return jobs from that number in the list of sorted jobs. Default is `1`, the first jobs.
- **number** (*integer*) – Return at most that number of jobs. Default returns the first 100 jobs.
- **sort** (*string*) – List of form `field (asc|desc)[,...]`. Sort by specific fields.
  - `jobId`
  - `type`
  - `state`
  - `user`
  - `startTime`
  - `priority`
- **user** (*boolean*) –
  - `true` (default) - Include only jobs created by current user
  - `false` - Include all jobs
- **field** (*string*) – Comma-separated list of fields to include in the result metadata.

**Produces**

- **application/xml, application/json** – JobListDocument
- **text/plain** – CRLF-delimited list of job ids.

**Role** _job_read

### Retrieve a job

**GET /job/** (*job-id*)
Return information about specified job.

When returning in format `text/plain`, only a string representation of the state is returned.

**Query Parameters**

- **metadata** (*boolean*) –
  - `true` - Include job metadata with all jobs.
  - `false` (default) - Do not include job metadata with all jobs.
- **field** (*string*) – Comma-separated list of fields to include in the result metadata.

**Status Codes**

- **404 Not found** – Invalid id

**Produces**

- **application/xml, application/json** – JobDocument
- **text/plain** – State of job

**Role** _job_read

### Search and count jobs

**PUT /job/search**
New in version 5.2.

Return jobs matching the given criteria and chosen facets for the result-set. There are three types of facets:

•`type` Counts occurrences of each *job type*

•`state` Counts occurrences of each *job state*

•`user` Counts occurrences of each user

**Query Parameters**

- **jobmetadata** (*string[]*) – Multiple query parameters can be specified. If no query parameters are specified, all jobs are returned.
  - *key* = *value* - Filter out only the jobs that has job metadata according to the filter criteria.

  Note: the metadata field `item` is generated and cannot be filtered on, instead the field `itemId` should be used.

  Note: = is part of the query parameter, and has to be encoded (`%3d`).
- **metadata** (*boolean*) –
  - `true` - Include job metadata with all jobs.
  - `false` (default) - Do not include job metadata with all jobs.

- **idonly** (*boolean*) –
  - `true` - Only return a list of ids
  - `false` (default) - Return job information such as job status
- **starttime-from** (*string*) – ISO 8601 timestamp. Return only jobs started after and at the given timestamp.
- **starttime-to** (*string*) – ISO 8601 timestamp. Return only jobs started before and at the given timestamp.
- **finishtime-from** (*string*) – ISO 8601 timestamp. Return only jobs finished after and at the given timestamp.
- **finishtime-to** (*string*) – ISO 8601 timestamp. Return only jobs finished before and at the given timestamp.
- **step** (*boolean*) –
  - `true` - Include step information in the job listing.
  - `false` (default) - Do not include step information in the job listing.
- **type** (*string*) – Comma-separated list of *job types*. Default is `all`, return all jobs.
- **state** (*string*) – Comma-separated list of *job states*. Default is `all`, return all jobs.
- **priority** (*string*) – Comma-separated list of *job priorities*. Default is `all`, return all jobs.
- **first** (*integer*) – Return jobs from that number in the list of sorted jobs. Default is `1`, the first jobs.
- **number** (*integer*) – Return at most that number of jobs. Default returns the first 100 jobs.
- **sort** (*string*) – List of form `field (asc|desc)[,...]`. Sort by specific fields.
  - `jobId`
  - `type`
  - `state`
  - `user`
  - `startTime`
  - `priority`
- **user** (*boolean*) –
  - `true` (default) - Include only jobs created by current user
  - `false` - Include all jobs
- **field** (*string*) – Comma-separated list of fields to include in the result metadata.

**Status Codes**

- **404 Not found** – Field not valid

**Produces**

- **application/xml, application/json** – JobListDocument
- **text/plain** – CRLF-delimited list of job ids.

**Accepts**

- **application/xml, application/json** – JobSearchDocument

**Role** _job_search

### Example

A faceted search should count the occurrences of each `type`, `state` and `user` for all existing jobs. It is possible to supply attributes `name`, `minCount`, `maxCount` and `maxResults`. Facets can be named to make it easier to distinguish between different facets. By supplying attributes `minCount`, `maxCount` any fields that has a count lower/higher than the specified values will be excluded. By default a minimum count of 0 and a maximum count of integer max size is returned. By default, all facet counts will be returned. By using the `maxResults` attribute, this behaviour can be changed.

```
PUT /job/search
Content-Type: application/xml

<JobSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <facet count="true" name="JobTypes">
    <field>type</field>
  </facet>
    <facet count="true" minCount="1" maxCount="5" maxResults="10">
      <field>state</field>
  </facet>
    <facet count="true">
      <field>user</field>
  </facet>
</JobSearchDocument>
```

```
<JobListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <hits>2</hits>
    <job>
      <jobId>VX-1</jobId>
      <user>admin</user>
      <started>2020-02-19T13:53:21.110Z</started>
      <finished>2020-02-19T13:53:21.242Z</finished>
      <status>FINISHED</status>
      <type>RAW_IMPORT</type>
      <priority>MEDIUM</priority>
    </job>
    <job>
      <jobId>VX-2</jobId>
      <user>admin</user>
      <started>2020-02-19T13:53:21.110Z</started>
      <finished>2020-02-19T13:53:21.242Z</finished>
      <status>STARTED</status>
      <type>AUTO_IMPORT</type>
      <priority>MEDIUM</priority>
    </job>
    <facet name="JobTypes">
      <field>type</field>
      <count fieldValue="RAW_IMPORT">1</count>
      <count fieldValue="AUTO_IMPORT">1</count>
    </facet>
    <facet>
      <field>state</field>
      <count fieldValue="FINISHED">1</count>
      <count fieldValue="STARTED">1</count>
```

```
    </facet>
    <facet>
      <field>user</field>
      <count fieldValue="admin">2</count>
    </facet>
</JobListDocument>
```

### Modify a job

**PUT /job/** (*job-id*)

Updates the job by setting job priority

**Query Parameters**

- **priority** (*string*) – Change the job priority. One of the valid *job priorities* <*job_priority*>: Default is `MEDIUM`

**Status Codes**

- **404 Not found** – Invalid id

**Role** _job_write

### Abort a job

**DELETE /job/** (*job-id*)

Does not delete the job, but aborts it.

To delete one or more jobs, use *DELETE /job*.

The job is marked for abortion, but the call may return before all tasks have been killed. Hence, the status return by this call is likely to be ABORTED_PENDING rather than ABORTED. Caller should poll the status of the job or use job notifications to find out when job has been fully aborted.

**Query Parameters**

- **reason** (*string*) – Reason for cancellation.
- **cleanup** (*boolean*) –
  - `true` (default) - Run cleanup steps before aborting.
  - `false` - Skip the cleanup steps, unless the job is already running. For READY jobs this means that the job will immediately be marked as ABORTED.

**Status Codes**

- **404 Not found** – Invalid id

**Role** _job_write

### Create a duplicate job

**POST /job/** (*job-id*) **/re-run**

Retrieves an existing job, duplicates it and starts the duplicated version.

**Query Parameters**

- **priority** (*string*) – The priority of the new job. If no priority is specified then the priority of the existing job will be used.

**Status Codes**

- **404 Not found** – Invalid id

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

### Start a job with custom type

**POST /job**

Starts a new job, of the type specified in the `type` parameter.

Changed in version 5.0.

Additional job metadata can also be added using an *optional* SimpleMetadataDocument. If any `jobmetadata` keys would collide between the query parameters and the SimpleMetadataDocument, the key and value from the SimpleMetadataDocument would have precedence over the query parameter.

**Query Parameters**

- **type** (*string*) – Required. The job type name.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Status Codes**

- **400** – Invalid job type name

**Accepts**

- **application/xml, application/json** – SimpleMetadataDocument

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

## 17.13.2 Job problem conditions

Jobs can enter the state WAITING if a recoverable problem has occurred. Depending on the problem the system might resolve itself or require manual assistance, e.g. out of storage space.

### List all job problems

**GET /job/problem**

Returns a list of unresolved problems, together with what jobs are waiting for them to be resolved.

**Produces**

- **application/xml, application/json** – JobProblemListDocument.

**Role** _job_read

#### List all problems for a job

**GET /job/** (*job-id*) **/problem**
Retrieves a list of problems that affects the specified job.

**Status Codes**

- **404 Not found** – Invalid id

**Produces**

- **application/xml, application/json** – JobProblemListDocument

**Role** _job_read

#### Delete one or several jobs

**DELETE /job**
Deletes a job and every related database entry.

**Query Parameters**

- **id** (*string*) – Required. Comma-separated list of job ids.

**Status Codes**

- **400 Invalid input** – The job is still running
- **404 Not found** – Invalid id

**Role** _job_write

### 17.13.3 Job states

#### Job states

The following states are defined for a job:

**READY** Job can be run, in queue.

**STARTED** One or several job steps are running.

**VIDINET_JOB** One or several job steps are running on Vidinet.

**FINISHED** The job has finished with success.

**FINISHED_WARNING** The job has finished, but a non-critical step failed.

**FAILED_TOTAL** The job has finished, but a critical Job step has failed.

**WAITING** The job is waiting for a condition.

**ABORTED_PENDING** A request to abort the job has been made.

**ABORTED** The job is aborted, and all job steps have finished.

#### Step states

The following states are defined for a job step (task):

**NONE** Job step has just been initialized. Normally this should not be returned.

**READY** Job step about to start.

**STARTED** Job step started, running in a transaction. (Short tasks.)

**STARTED_ASYNCHRONOUS** Job step started, running outside of an transaction. (Longer tasks, or tasks that cannot execute in a Java class.)

**STARTED_PARALLEL** Job step has started, and signals that other parallel tasks can start.

**STARTED_PARALLEL_ASYNCHRONOUS** Job step has started, and signals that other parallel tasks can start.

**STARTED_SUBTASKS** Job step has started, and will be performed in multiple subtasks.

**FINISHED** Job step has finished successfully.

**FAILED_RETRY** Job step has failed, but will be retried.

**FAILED_FATAL** Job step has failed, and will not be retried.

**WAITING** Job step is waiting for a condition, see *List all problems for a job* for cause.

**DISAPPEARED** The job worker was missing (possible cause is a restart of the application server). The job step will be re-run.

### 17.13.4 Job priority

Jobs have a priority setting that determines their order of execution. The following priority levels exists, from lowest to highest: `LOWEST`, `LOW`, `MEDIUM`, `HIGH`, `HIGHEST`, and `IMMEDIATE`.

> **Warning:** Jobs with priority `IMMEDIATE` are always started, even if the *max concurrent jobs* limit is reached. This could impact system performance. To execute the job with `IMMEDIATE` priority the user must be a super user, that is, have role `_super_access_user`.

### 17.13.5 Job types

The following job types exists. They can also be retrieved using *`GET /jobtype`*.

**NONE** Not used.

**IMPORT** Not used.

**PLACEHOLDER_IMPORT** Regular import (using URI or file id to existing or new item).

**RAW_IMPORT** Import where essence is in request body.

**AUTO_IMPORT** Import using auto import rules.

**SHAPE_IMPORT** Import using URI or file id to a shape.

**SIDECAR_IMPORT** Import a sidecar file to an existing item.

**ESSENCE_VERSION** Import a new essence version.

**TRANSCODE** Transcode of item.

**TRANSCODE_RANGE** Transcode of part of item.

**CONFORM** Conform an item sequence.

**TIMELINE** Conform a timeline.

**THUMBNAIL** Create thumbnails or posters of item.

**ANALYZE.** Do analysis of item.

**SHAPE_UPDATE** Update shape information of item.

**RAW_TRANSCODE.** Send transcode job directly to transcoder.

**EXPORT** Export item to remote location.

**COPY_FILE** Copy file (and keep track of new copy).

**MOVE_FILE** Move file.

**DELETE_FILE** Delete file.

**LIST_ITEMS** Generate item report.

**FILE_ANALYZE** Analyze a file to deduce its shape.

**IMF_ANALYZE** Analyze an IMF package to deduce its shape.

#### List all job types

**GET /jobtype**
Get list of job types

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – list of job types

### 17.13.6 Job metadata

Additional job metadata can be specified using the `jobmetadata` parameter, when a job is created. Note that the equals sign is part of the value of the query parameter, so it has to be URL encoded (%3d).

---

**Hint:** Prefer to always prefix any custom job metadata by the name of your application, for example, `myApp_customSetting`, to avoid conflict with any existing or future job metadata used by Vidispine.

---

#### Reserved keys

**smpteTimeCode**
The first frame to be included in transcoded output.

**Type** String (SMPTE timecode)

**lastSmpteTimeCode**
The last frame to be included in transcoded output.

**Type** String (SMPTE timecode)

**checksumMode**
Can be set to `transfer` to have the checksum computed during the transfer step of the import job.

**Note** This will not work if the files are transferred by the transcoder.

**cerifyPriority**
The priority to assign jobs created in Cerify. One of `LOW`, `MEDIUM` or `HIGH`.

**Default** `LOW`

## 17.14 Libraries

A library can be seen as a lightweight collection that is deleted on a regular basis if it is not being used. Libraries can only contain items.

### 17.14.1 Managing libraries

#### List all libraries

**GET /library**

Retrieves a list of the ids of all known libraries.

**Query Parameters**

- **first** (*integer*) – From the resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** (*integer*) – The number of entities to fetch. Default is `100`.
- **autoRefresh** (*boolean*) – Only list libraries with the specified auto refresh status.
- **frequencyFrom** (*integer*) – Only list libraries whose update frequency is greater than it.
- **frequencyTo** (*integer*) – Only list libraries whose update frequency is less than it.
- **updateMode** (*string*) – Only list libraries with the specified update mode.

**Produces**

- **application/xml, application/json** – URIListDocument containing the ids of all the libraries.
- **text/plain** – CRLF-delimited list of ids

**Role** _library_read

#### Example

```
GET /library
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <uri>VX*48</uri>
   <uri>VX*49</uri>
   <uri>VX*45</uri>
</URIListDocument>
```

#### Create a library

**POST /library**

Creates a library and returns the id of the created library.

**Query Parameters**

- **externalId** (*string*) – An external identifier to assign to the library.

**Accepts**

- **application/xml, application/json** – ItemListDocument that contains the ids of any items that should be added to the library

**Produces**

- **application/xml, application/json** – URIListDocument containing the id of the created library.
- **text/plain** – CRLF-delimited list of ids

**Status Codes**

- **400** – If the external id is already in use.

**Role** _library_write

### Example

```
POST /library
Content-Type: application/xml

<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item id="VX-250"/>
   <item id="VX-1000"/>
</ItemListDocument>
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <uri>VX*48</uri>
</URIListDocument>
```

### Delete a library

**DELETE /library/** (*library-id*)

Deletes the library with the specified id.

**Query Parameters**

- **async** (*boolean*) –
  - `true` - Start a DELETE_LIBRARY job that removes the library.
  - `false` (default) - Remove the library immediately.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument containing library delete job.

**Status Codes**

- **202** – The library will be removed by the returned job.
- **200** – The library has been deleted.

**Role** _library_write

### Example

```
DELETE /library/VX*51
```

### Delete multiple libraries

**DELETE /library**

Deletes the libraries with the specified ids.

**Query Parameters**

- **id** (*string*) – Required. Comma-separated list of library ids or external ids.
- **async** (*boolean*) –
  - `true` - Start a DELETE_LIBRARY job that removes the libraries.
  - `false` (default) - Remove the libraries immediately.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument containing library delete job.

**Status Codes**

- **202** – The libraries will be removed by the returned job.
- **200** – The libraries have been deleted.

**Role** _library_write

#### Example

```
DELETE /library?id=VX*50,VX*51
```

### 17.14.2 Library settings

#### Retrieve library settings

**GET /library/** (*library-id*) **/settings**
Retrieves the settings and status of a library.

**Produces**

- **application/xml, application/json** – LibrarySettingsDocument

**Role** _library_read

#### Example

```
GET /library/VX*67/settings HTTP/1.1
```

```
<LibrarySettingsDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <id>VX*67</id>
   <username>admin</username>
   <updateMode>REPLACE</updateMode>
   <autoRefresh>true</autoRefresh>
   <query>
      <field>
      <name>originalWidth</name>
      <range>
```

```
          <value>640</value>
          <value>720</value>
        </range>
        </field>
    </query>
</LibrarySettingsDocument>
```

#### Update library settings

**PUT /library/** (*library-id*) **/settings**
Update the settings of a library.

**Accepts**

- **application/xml, application/json** – LibrarySettingsDocument

**Role** _library_write

### 17.14.3 Library content

#### Retrieve library content

**GET /library/** (*library-id*)
Returns the items together with any requested data.

**Query Parameters**

- **starttc** (*boolean*) –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.
- **first** (*integer*) – From the resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** (*integer*) – The number of entities to fetch. Default is `100`.
- **count** (*boolean*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **noauth-url** (*boolean*) –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **baseURI** (*string*) – Which base URI to use for the thumbnail URLs.
- **p** (*string*) – Comma-separated list of *paths* specifying the content to include. Overrides the content and filter parameters.
- **content** (*string*) – Comma-separated list of the types of content to retrieve. Valid values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.

  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **`field`** (*`string`*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **`group`** (*`string`*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **`language`** (*`string`*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **`sampleRate`** (*`string`*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **`track`** (*`string`*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 − t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **`include`** (*`string`*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **`includeValues`** (*`boolean`*) – Return the value enumeration for each metadata field.
- **`conflict`** (*`string`*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **`terse`** (*`string`*) –
  - `yes` - Return metadata in *terse format*.
  - `no` (default) - Return metadata in verbose format.

- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** (*boolean*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.
- **revision** (*string*) – Specifying which metadata revision to display. Only used if requesting a single item or collection.
- **mergedType** (*string*) – The type of operation to check for.
- **mergedPermission** (*string*) – The lowest required permission level.
- **mergedExtradata** (*string*) – Any possible extra data.
- **uriType** (*string*) – Comma-separated list of format types (container format) to return.
- **scheme** (*string*) – URI scheme to return.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodType** (*string*) – *Access method*.
  - `AUTO` - Gives an APInoauth URI to the media. Access to file is tunneled through Vidispine.
  - `AZURE_SAS` - If the storage schema is azure:// you can get direct access to the media. The resulting URI will not tunnel through Vidispine but rather point directly to the media location at the azure storage.
- **methodMetadata** (*string*) – *Metadata* used with storage method.
- **tag** (*string*) – A *URI parameter*: Comma-separated list of *shape tags* to return.
- **version** (*string*) – Specifying which essence version to return for shapes. If special value `all`, display all versions. If special value `latest` (default), display latest version of shapes.
- **closedFiles** (*boolean*) – A *URI parameter*:
  - `true` (default) - Return only URIs that point to closed files.
  - `false` - Return all URIs.
- **storage** (*string[]*) – List of storage ids. Return only files from specific storages. Can be specified multiple times.
- **storageGroup** (*string*) – Storage group id. Return only files from storages specified in the storage group.

**Produces**

- **application/xml, application/json** – ItemListDocument containing the items together with the requested data.

**Role** _library_read

**Role** _metadata_read (`content=metadata`)

**Role** _item_uri (`content=uri`)

**Role** _thumbnail_read (`content=poster` and `content=thumbnail`)

**Role** _access_control_read (`content=access` and `content=merged-access`)

**Role** _item_id_read (`content=external`)

### Example

```
GET /library/VX*48/?content=access HTTP/1.1
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item id="VX-250">
      <access>
         <type>GENERIC</type>
         <permission>ALL</permission>
      </access>
      <access>
         <type>METADATA</type>
         <permission>ALL</permission>
      </access>
      <access>
         <type>ID</type>
         <permission>ALL</permission>
      </access>
      <access>
         <type>URI</type>
         <permission>ALL</permission>
      </access>
   </item>
   <item id="VX-1000">
      <access>
         <type>GENERIC</type>
         <permission>ALL</permission>
      </access>
      <access>
         <type>METADATA</type>
         <permission>ALL</permission>
      </access>
      <access>
         <type>ID</type>
         <permission>ALL</permission>
      </access>
      <access>
         <type>URI</type>
         <permission>ALL</permission>
      </access>
   </item>
</ItemListDocument>
```

### Add multiple items to a library

**PUT /library/** (*library-id*)
Adds all the items specified in the document to the library.

**Accepts**

- **application/xml, application/json** – ItemListDocument that contains the item ids.

**Role** _library_write

### Example

```
PUT /library/VX*48
Content-Type: application/xml

<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item id="VX-1000"/>
   <item id="VX-250"/>
</ItemListDocument>
```

```
200 OK
```

### Add an item to a library

**PUT /library/** (*library-id*) **/**
*item-id* Adds the specified item to the library.

**Role** _library_write

### Example

```
PUT /library/VX*48/VX-251
```

```
200 OK
```

### Remove an item from a library

**DELETE /library/** (*library-id*) **/**
*item-id* Removes the specified item from the library.

**Role** _library_write

### Example

```
DELETE /library/VX*48/VX-251
```

```
200 OK
```

### Modify metadata of the items in a specific library

**PUT /library/** (*library-id*) **/item-metadata**
Modify metadata of the items in a specific library

**Query Parameters**

- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Accepts**

- **application/xml, application/json** – MetadataDocument the metadata to apply to the items.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _library_write

### 17.14.4 Listing library items in batch

#### Creating an item list job

**POST /library/** (*library-id*) **/list**

Starts a new job that goes through all the items in the specific library and outputs a file to the supplied URI.

The output format depends on the specified parameter, if set to XML an ItemListDocument will be produced. Furthermore if an XSLT is given the ItemListDocument will be transformed.

**Query Parameters**

- **destinationUri** (*string*) – Required. The URI to output the CSV file to.
- **outputFormat** (*string*) – Specifies the output format. One of `xml` (default) and `csv`.
- **field** (*string*) – Comma-separated list of metadata fields to include in the result. Default is `title`
- **data** (*string*) – Specifies any additional data that should be included with the metadata fields.
- **p** (*string*) – Comma-separated list of *paths* specifying the content to include. Overrides the field and data parameters. Only supported for XML output.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Accepts**

- **application/xslt** – An optional XSLT capable of transforming ItemListDocument.

**Produces**

- **application/xml, application/json** – JobDocument.

**Role** _library_read

#### Example

```
POST /library/VX*75/list?p=id,shape.containerComponent.duration&destinationUri=file:/
↪home/user/output.xml
Content-Type: application/xml

<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-121</jobId>
  <user>admin</user>
  <started>2016-02-21T10:11:42.998+01:00</started>
  <status>READY</status>
  <type>LIST_ITEMS</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

```
$ xmllint --format /home/user/output.xml
```

```
<?xml version="1.0"?>
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <item xmlns="http://xml.vidispine.com/schema/vidispine" id="VX-47">
    <shape>
      <containerComponent>
        <duration>
          <samples>1871720000</samples>
          <timeBase>
            <numerator>1</numerator>
            <denominator>1000000</denominator>
          </timeBase>
        </duration>
      </containerComponent>
    </shape>
  </item>
  <item xmlns="http://xml.vidispine.com/schema/vidispine" id="VX-48">
    <shape>
      <containerComponent>
        <duration>
          <samples>1871763000</samples>
          <timeBase>
            <numerator>1</numerator>
            <denominator>1000000</denominator>
          </timeBase>
        </duration>
      </containerComponent>
    </shape>
  </item>
</ItemListDocument>
```

### Re-index autorefresh index

**PUT /library/re-index**
Clears and rebuilds the separate index for autorefresh library queries in Elasticsearch. Requests are ignored for Solr.

**Produces**

- **text/plain** – re-indexing status.

**Role** _library_write

### Re-index specific autorefresh library

**PUT /library/** (*library-id*) **/re-index**
Clears and re-adds the library queries indexed in Elasticsearch. Requests are ignored for Solr, non-autorefreshing libraries, and libraries with update mode `TRANSIENT`.

**Produces**

- **text/plain** – re-indexing status.

**Role** _library_write

## 17.15 License

The license resource allows you to view and update your Vidispine license. It is also the entry point to use if the system is being used as a licensing provider.

### 17.15.1 Version and license

#### Retrieve version and license information

**GET /version**
Display your license allowance and current system usage.

The `systemInfo` element in the response shows the MAC addresses discovered on the local system. The MAC-address(es) in the license key must match that/those of your system.

**Produces**

- **application/xml, application/json** – VersionDocument
- **text/plain** – The version details in a informational text format.

#### Retrieve the license file

**GET /license**
Retrieves the contents of the installed license file.

**Produces**

- **text/plain** – The contents of the license file.

### 17.15.2 Slave management and monitoring

Deprecated since version 5.0: A license file should be used instead.

#### Install a slave license on a master node

Deprecated since version 5.0.

**PUT /license/slave**
Installs the slave license file with the given path.

**Query Parameters**

- **path** (*string*) – Required. The name of the slave license file.

**Produces**

- **application/xml, application/json** – SlaveLicenseDocument

#### Install a slave license on a master node

Deprecated since version 5.0.

**POST /license/slave**
Installs a slave license.

**Accepts**

- **text/plain** – The content of the slave license file.

**Produces**

- **application/xml, application/json** – SlaveLicenseDocument

### List all slaves

**GET /license/slave**
Returns a list of all the slave nodes connected to this master. Slaves that have not been seen for more than 180 minutes will not be available.

**Produces**

- **application/xml, application/json** – SlaveListDocument

### List slave license status

**GET /license/slave/** (*id*)
Returns information about the slave with the given id.

**Produces**

- **application/xml, application/json** – VersionDocument

### Retrieve a slave license file

**GET /license/slave/** (*id*) **/license**
Returns the slave license for a specific slave.

**Produces**

- **application/xml, application/json** – SlaveLicenseDocument

### Delete a slave instance

Deprecated since version 5.0.

**DELETE /license/slave/** (*id*)
Removes the slave with the given id.

### List all installed slave licenses

Deprecated since version 5.0.

**GET /license/slave/license**
Returns the `id` and `SlaveIdentifier` of all installed slave license on a master

**Query Parameters**

- **slaveId** (*string*) – Find the slave license with this slave id.

**Produces**

- **application/xml, application/json** – SlaveLicenseListDocument

### List installed slave licenses by id

Deprecated since version 5.0.

**GET /license/slave/license/** (*id*)
Returns the detail of an installed slave license with the given id

**Produces**

- **application/xml, application/json** – SlaveLicenseDocument
- **text/plain** – The slave license file.

#### Install or update slave connection string

**PUT /APInoauth/license/auth-info**

**Accepts**

- **application/xml, application/json** – SlaveAuthInfoDocument

#### Example

```
PUT /APInoauth/license/auth-info
Content-Type: application/xml

<SlaveAuthInfoDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <masterHost>http://192.168.0.1:8080/</masterHost>
  <masterHost>http://my.other.server:8080/</masterHost>
  <slaveId>your-slave-id</slaveId>
</SlaveAuthInfoDocument>
```

License validity and status can be seen from *GET /version*.

## 17.16 Metadata

### 17.16.1 Auto-projection rules

Automatic projection rules.

#### Working with automatic projection rules

#### List all automatic projection rules

**GET /auto-projection**
Retrieves all automatic projection rules.

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – CRLF-delimited list of names

**Role** _auto_projection_read

#### List all disabled automatic projection rules

**GET /auto-projection/disable**
Retrieves all disabled automatic projection rules.

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – CRLF-delimited list of names

**Role** _auto_projection_read

#### List all enabled automatic projection rules

**GET /auto-projection/enable**
Retrieves all enabled automatic projection rules.

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – CRLF-delimited list of names

**Role** _auto_projection_read

### Retrieve an automatic projection rule

**GET /auto-projection/** (*name*)
Retrieves a specific projection rule.

**Produces**

- **application/xml, application/json** – AutoProjectionRuleDocument

**Role** _auto_projection_read

### Create an automatic projection rule

**PUT /auto-projection/** (*name*)
Creates a new projection rule based on the information in the AutoProjectionRuleDocument.

**Accepts**

- **application/xml, application/json** – AutoProjectionRuleDocument

**Role** _auto_projection_write

### Example

```
PUT /auto-projection/testProjection
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AutoProjectionRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <step>
    <order>1</order>
    <description>step1 description</description>
    <script>...</script>
  </step>
  <step>
    <order>2</order>
    <description>step2 description</description>
    <script>...</script>
  </step>
  <description>rule description</description>
  <inputFilters>
    <inputFilter>oldMetadata</inputFilter>
    <inputFilter>shapeDocument</inputFilter>
    <bulkyMetadataKeysRegex>.*</bulkyMetadataKeysRegex>
  </inputFilters>
  <triggers>
    <trigger>itemMetadata</trigger>
    <trigger>shapeMetadata</trigger>
  </triggers>
</AutoProjectionRuleDocument>
```

#### Delete an automatic projection rule

**DELETE /auto-projection/** (*name*)
Deletes the automatic projection rule.

**Role** _auto_projection_write

#### Disable an automatic projection rule

**PUT /auto-projection/** (*name*) **/disable**
Disables the automatic projection rule.

**Role** _auto_projection_write

#### Enable an automatic projection rule

**PUT /auto-projection/** (*name*) **/enable**
Enables the automatic projection rule.

**Role** _auto_projection_write

### 17.16.2 Bulky metadata

Bulky metadata can be used to store large amounts of timed metadata. The metadata is arranged in a key-value fashion, where the key is the quintuple (*field*, start, end, stream, channel).

The metadata is not indexed, but is typically used as a temporary container for metadata that is to be processed later, for example using *auto-projection rules*.

#### Managing bulky metadata

Both items and shapes can hold bulky metadata.

#### Insert values in bulk

**PUT /item/** (*item-id*) **/metadata/bulky/**
**PUT /item/** (*item-id*) **/shape/**
*shape-id***/metadata/bulky/** Inserts all key-value pairs from a given document.

**Accepts**

- **application/xml, application/json** – BulkyMetadataDocument

**Role** _metadata_write

#### Example

```
PUT /item/VX-250/metadata/bulky
Content-Type: application/xml

<BulkyMetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <field start="3" end="5">
      <key>mykey</key>
      <value>This is the value of mykey for the first interval.</value>
   </field>
   <field start="5" end="9">
      <key>mykey</key>
      <value>This is the value of mykey for the second interval.</value>
```

```
   </field>
</BulkyMetadataDocument>
```

### Retrieve all keys used in the bulky metadata

**GET /item/** (*item-id*) **/metadata/bulky/**
**GET /item/** (*item-id*) **/shape/**
*shape-id***/metadata/bulky/** Retrieves a list of all keys in the bulky metadata.

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – CRLF-delimited list of keys

**Role** _metadata_read

### Delete all keys used in the bulky metadata

**DELETE /item/** (*item-id*) **/metadata/bulky/**
**DELETE /item/** (*item-id*) **/shape/**
*shape-id***/metadata/bulky/** Removes all the keys in the bulky metadata.

**Role** _metadata_write

### Read values

**GET /item/** (*item-id*) **/metadata/bulky/**
*key*
**GET /item/** (*item-id*) **/shape/**
*shape-id***/metadata/bulky/***key* Retrieves all values of a certain key over a specified interval. All values for that key can be retrieved by specifying start as "-INF" and end as "+INF".

**Query Parameters**

- **start** (*string*) – A *time code* that defines the start of the interval.
- **end** (*string*) – A *time code* that defines the end of the interval.

**Produces**

- **application/xml, application/json** – BulkyMetadataDocument

**Role** _metadata_read

### Example

```
GET /item/VX-123/metadata/bulky/mykey?start=-INF&end=+INF
```

```
<BulkyMetadataDocument id="VX-123" xmlns="http://xml.vidispine.com/schema/vidispine">
   <field start="3" end="5">
      <key>mykey</key>
      <value>This is the value of mykey for the first interval.</value>
   </field>
   <field start="5" end="9">
      <key>mykey</key>
      <value>This is the value of mykey for the second interval.</value>
   </field>
</BulkyMetadataDocument>
```

### Returning bulky metadata as a file

**GET /item/** (*item-id*) **/metadata/bulky/** *key***/as-file**
**GET /item/** (*item-id*) **/shape/** *shape-id***/metadata/bulky/** *key***/as-file** Bulky metadata that is stored as a map and that has the keys *filename* and *content* can be retrieved as a file. If the map contains several entries with those keys, the first one will be returned. To have more granular control over which entry is returned, the query parameters below can be use to select which entry to match on.

**Query Parameters**

- **start** (*string*) – A *time code* that defines the start of the interval.
- **end** (*string*) – A *time code* that defines the end of the interval.
- **stream** (*integer*) – The stream index.
- **channel** (*integer*) – The audio channel index.
- **itemTrack** (*string*) – The track in the item.
- **extraMapValues** (*string*) – Additional map values to filter on in the format *key*=*value*. This parameter can be used multiple times.

**Produces**

- **application/octet-stream** – The binary file content

**Role** _metadata_read

### Example

```
<BulkyMetadataDocument id="VX-123" xmlns="http://xml.vidispine.com/schema/vidispine">
   <field start="-INF" end="+INF">
      <key>mykey</key>
      <maps>
        <map>
          <entry key="filename">my_pdf.pdf</entry>
          <entry key="content">...</entry>
          <entry key="type">PDF</entry>
        </map>
        <map>
          <entry key="filename">my_xml.xml</entry>
          <entry key="content">...</entry>
          <entry key="type">XML</entry>
        </map>
     </maps>
   </field>
   <field start="10" end="30">
      <key>mykey</key>
      <maps>
        <map>
          <entry key="filename">another_pdf.pdf</entry>
          <entry key="content">...</entry>
          <entry key="type">PDF</entry>
        </map>
        <map>
          <entry key="filename">another_xml.xml</entry>
          <entry key="content">...</entry>
          <entry key="type">XML</entry>
```

```
        </map>
      </maps>
   </field>
</BulkyMetadataDocument>
```

If we want to retrieve the "my_pdf.pdf" entry as a binary file, we can use the following request (note that the = sign in the *extraMapValues* parameter is URL encoded in this example):

```
GET /item/VX-123/metadata/bulky/mykey/as-file?start=-INF&end=+INF&extraMapValues=type
↪%3DPDF
```

But the follwing request would also get the same file:

```
GET /item/VX-123/metadata/bulky/mykey/as-file?start=-INF&end=+INF&
↪extraMapValues=filename%3Dmy_pdf.pdf
```

### Insert values

**PUT /item/** (*item-id*) **/metadata/bulky/**
*key*
**PUT /item/** (*item-id*) **/shape/**
*shape-id***/metadata/bulky/***key* Inserts a value at the specified interval for the given key. If the key already has a value at that specific interval then that value will be overwritten.

**Query Parameters**

- **start** (*string*) – A *time code* that defines the start of the interval.
- **end** (*string*) – A *time code* that defines the end of the interval.
- **stream** (*integer*) – The stream index.
- **channel** (*integer*) – The audio channel index.
- **itemTrack** (*string*) – The track in the item.

**Accepts**

- **text/plain** – The value to set.

**Role** _metadata_write

### Example

```
PUT /item/VX-123/metadata/bulky/mykey?start=3&end=5
Content-Type: text/plain

This is the value of mykey for the first interval.
```

```
PUT /item/VX-123/metadata/bulky/mykey?start=5&end=9
Content-Type: text/plain

This is the value of mykey for the second interval.
```

### Remove values

**DELETE /item/** (*item-id*) **/metadata/bulky/**
*key*

**DELETE /item/** (*item-id*) **/shape/**
*shape-id***/metadata/bulky/***key** Removes all the values for a certain key over the specified interval.

**Query Parameters**

- **start** (*string*) – A *time code* that defines the start of the interval.
- **end** (*string*) – A *time code* that defines the end of the interval.

**Role** _metadata_write

### 17.16.3 Global metadata

Global metadata is metadata that is not associated with any item or collection. It can primarily be used as a reference, for example holding a field that is referenced from many items.

#### Retrieve the global metadata

**GET /metadata**
Retrieves the global metadata.

**Query Parameters**

- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 − t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.

- **language** (*string*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **samplerate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **conflict** (*string*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **include** (*string*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **includeConstraintValue** (*string*) – Comma-separated list of fields whose "display value" should be retrieved from the *metadata dataset*.
  - `all` (default) - Return the "display value" of all fields.
  - `none` - No "display value" will be returned. The fields will only have `id` set.
  - *comma-separated field names* - Return the "display value" of the specified fields.

**Produces**

- **application/xml, application/json** – MetadataDocument

**Role** _metadata_global_read

### Update the global metadata

**PUT /metadata**

Modifies the global metadata. This resource shares the same query parameters as the item metadata resource.

**Query Parameters**

- **revision** (*string*) – The known revision. If not specified, the change set will attempt to override existing change sets.
- **skipForbidden** (*boolean*) – Skip fields or groups that the user doesn't have write access to. Default is `false`
- **onlyReturnChanges** (*boolean*) – New in version 4.16.6.
  - `true` - Only return the changed entries.
  - `false` (default) - Return the whole global metadata after the update.

**Accepts**

- **application/xml, application/json** – MetadataDocument

**Produces**

- **application/xml, application/json** – MetadataDocument

### Retrieve metadata by UUID

**GET /metadata/**(*uuid*)

Retrieves the metadata entry that matches the UUID.

**Produces**

- **application/xml, application/json** – MetadataEntryDocument

**Role** _metadata_global_read

### Example

```
GET /metadata/c3dc7918-9316-4fef-b4fc-ff2b0149e854
```

```
<MetadataEntryDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field uuid="c3dc7918-9316-4fef-b4fc-ff2b0149e854" user="system" timestamp="2011-01-
↪10T10:00:54.845+01:00" change="VX-7">
    <name>originalVideoCodec</name>
    <value uuid="199255d8-59ec-421e-9c7b-757c46c92b14" user="system" timestamp="2011-
↪01-10T10:00:54.845+01:00" change="VX-7">h264</value>
  </field>
</MetadataEntryDocument>
```

```
GET /metadata/199255d8-59ec-421e-9c7b-757c46c92b14
```

```
<MetadataEntryDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <value uuid="199255d8-59ec-421e-9c7b-757c46c92b14" user="system" timestamp="2011-01-
↪10T10:00:54.845+01:00" change="VX-7">h264</value>
</MetadataEntryDocument>
```

### Remove metadata by UUID

**DELETE /metadata/**(*uuid*)

Removes the metadata with the specified UUID.

**Role** _metadata_global_write

### Example

```
DELETE /metadata/6fba17bb-ed52-43ab-86b7-07f5494edeed
```

```
200 OK
```

## 17.16.4 Document metadata

Document metadata is similar to global metadata but instead of having a single, large global metadata document, it could be spread to multiple, small documents. This to reduce the size of the metadata and to improve performance.

### Managing documents

### List all documents

**GET /document**

Retrieves the list of metadata documents.

**Query Parameters**

- **first** (*integer*) – Return documents from that number in the document list. Default is `1`.
- **number** (*integer*) – Return at most that number of documents. Default is `100`.

**Produces**

- **application/xml, application/json** – *DocumentListDocument*

**Role** _document_read

### Example

```
GET /document
```

```
<?xml version="1.0" ?>
<DocumentListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <document>
        <name>producer</name>
        <uri>http://localhost:8080/API/document/producer</uri>
    </document>
    <document>
        <name>editor</name>
        <uri>http://localhost:8080/API/document/editor</uri>
    </document>
</DocumentListDocument>
```

### Retrieve a document

**GET /document/** (*name*)

Retrieves the document with the specified name. This resource shares the same query parameters as the *item metadata resource*.

**Query Parameters**

- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.

- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1* − *t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **language** (*string*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **samplerate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **conflict** (*string*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **include** (*string*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.

**Produces**

- **application/xml, application/json** – MetadataDocument

**Role** _document_read

### Example

```
GET /document/editor
```

```
<?xml version="1.0" ?>
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <revision>VX-20383</revision>
```

```
    <timespan end="+INF" start="-INF">
        <field change="VX-20383" timestamp="2014-11-18T10:29:45.166+01:00" user="admin
↪" uuid="ff48980c-6431-4234-83ac-bd8d0af9462d">
            <name>editor_name</name>
            <value change="VX-20383" timestamp="2014-11-18T10:29:45.166+01:00" user=
↪"admin" uuid="5b9929ba-cce9-43eb-b63b-62a2426b9891">Bob</value>
        </field>
    </timespan>
</MetadataDocument>
```

### Update or creates a document

**PUT /document/** (*name*)

Creates a new or modifies the existing document with the specified name.

**Query Parameters**

- **revision** (*string*) – The known revision. If not specified, the change set will attempt to override existing change sets.
- **skipForbidden** (*boolean*) – Skip fields or groups that the user doesn't have write access to. Default is `false`.

**Accepts**

- **application/xml, application/json** – MetadataDocument

**Produces**

- **application/xml, application/json** – MetadataDocument

**Role** _document_write

### Example

```
PUT /document/editor
```

```
<?xml version="1.0" encoding="UTF-8"?>
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    <field>
      <name>editor_name</name>
      <value>Bob</value>
    </field>
  </timespan>
</MetadataDocument>
```

Response:

```
<?xml version="1.0" ?>
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <revision>VX-20424</revision>
    <timespan end="+INF" start="-INF">
        <field change="VX-20424" timestamp="2014-11-18T13:36:46.130+01:00" user="admin
↪" uuid="6279f922-621b-4f0a-b09c-41586736eb9e">
            <name>title</name>
            <value change="VX-20424" timestamp="2014-11-18T13:36:46.130+01:00" user=
↪"admin" uuid="0c662ed2-6623-45ac-9272-b6abc232488e">Bob</value>
        </field>
```

```
    </timespan>
</MetadataDocument>
```

### Search for documents

**PUT /document**

Searches for documents matching a provided ItemSearchDocument.

Note that document indexing is disabled by default. Make sure to enable document indexing using *indexDocumentMetadata* and *trigger a reindex* so that any existing documents are added to the index.

New in version 5.0.

**Query Parameters**

- **first** (*integer*) – Start returning results from this index. Default is `1`.
- **number** (*integer*) – Return at most this number of documents. Default is `100`.
- **count** (*boolean*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.

**Accepts**

- **application/xml, application/json** – ItemSearchDocument

**Produces**

- **application/xml, application/json** – SearchResultDocument

### View change sets

**GET /document/**(*name*)**/changes**

Retrieves all change sets that have been applied to the document.

**Query Parameters**

- **change** (*string*) – Retrieve a single change set.
- **first** (*integer*) – Return change sets from that number in the list of sorted change sets. Default is `1`.
- **number** (*integer*) – Return at most that number of change sets. Default is all change sets.

**Response Headers**

- **Link** – Contains URLs to the previous, next, first and last pages.

**Produces**

- **application/xml, application/json** – MetadataChangeSetDocument

**Role** _document_read

### Example

```
GET /document/editor/changes
```

```
<?xml version="1.0" ?>
<MetadataChangeSetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <changeSet>
        <id>VX-20381</id>
        <metadata>
            <revision>VX-20380</revision>
            <timespan end="+INF" start="-INF">
                <field change="VX-20381" timestamp="2014-11-18T10:27:31.192+01:00"␣
↪user="admin" uuid="7d1032a5-2ced-42ce-a553-1fbbaf2faeef">
                    <name>editor_name</name>
                    <value change="VX-20381" timestamp="2014-11-18T10:27:31.192+01:00
↪" user="admin" uuid="c0426f6f-f1f0-4729-aaef-43a7f3b5bbfa">alice</value>
                </field>
            </timespan>
        </metadata>
    </changeSet>
    <changeSet>
        <id>VX-20382</id>
        <metadata>
            <revision>VX-20381</revision>
            <timespan end="+INF" start="-INF">
                <field change="VX-20382" timestamp="2014-11-18T10:27:31.229+01:00"␣
↪user="admin" uuid="7d1032a5-2ced-42ce-a553-1fbbaf2faeef">
                    <name>editor_name</name>
                    <value change="VX-20382" timestamp="2014-11-18T10:27:31.229+01:00
↪" user="admin" uuid="281d6f83-a624-4fa0-81fd-debe0e8bdf68">Bob</value>
                </field>
            </timespan>
        </metadata>
    </changeSet>
</MetadataChangeSetDocument>
```

#### Delete a document

**DELETE /document/**(*name*)

Removes the metadata document with the specified name.

**Role** _document_write

### 17.16.5 Key-value metadata

Some resources support basic key-value metadata (not to be confused with *item metadata*). This metadata is not indexed nor is it revision controlled. The role required to use the metadata depends on the resource.

Keys can be hierarchical, meaning the key can include the slash character.

Changed in version 4.16: Support for key-value metadata was added to shape-tags, libraries and task-definitions.

#### Supported resources

The resources that support key-value metadata, and the roles required to read and write to them can be seen in the table below. These are referred to as `{key-value-metadata-resources}`.

| Resource | URI | Read role | Write role |
|---|---|---|---|
| *Groups* | /group/{group-name}/metadata | _group_read | _group_write |
| *Users* | /user/{username}/metadata | _admininstrator (or none for user's own data) | _administrator (or _user_metadata_write for user's own data) |
| *Storages* | /storage/{storage-id}/metadata | _storage_read | _storage_write |
| *Storage methods* | /storage/{storage-id}/method/{method-id}/metadata | _storage_read | _storage_write |
| *Storage groups* | /storage/storage-group/{storage-group-name}/metadata | _storage_read | _storage_group_write |
| *Metadata fields* | /metadata-field/{field-name}/metadata | _metadata_field_read | _metadata_field_write |
| *Metadata field groups* | /metadata-field/field-group/{group-name}/metadata | _metadata_field_read | _metadata_field_group_write |
| *Shapes* | /item/{item-id}/shape/{shape-id}/metadata | _shape_read | _shape_write |
| *Shape components* | /item/{item-id}/shape/{shape-id}/component/{component-id}/metadata | _shape_read | _shape_write |
| *Files* | /storage/{storage-id}/file/{file-id}/metadata | _file_read | _file_write |
| *Task groups* | /task-group/{groupname}/metadata | _admininstrator | _admininstrator |
| *Libraries* | /library/{library-id}/metadata | _library_read | _library_write |
| *Shape tags* | /shape-tag/{tag-name}/metadata | _shape_tag_read | _shape_tag_write |
| *Task definitions* | /task-definition/{task-id}/metadata | _taskdefinition_read | _taskdefinition_write |

### Managing key-value metadata

### Retrieve all metadata

**GET {key-value-metadata-resource}**
Retrieves all key-value pairs associated with the specified entity.

**Produces**

- **application/xml, application/json** – A SimpleMetadataDocument containing all key-value pairs.

### Example

```
GET /user/myuser/metadata
```

```
<SimpleMetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <key>occupation</key>
    <value>developer</value>
  </field>
  <field>
    <key>location</key>
    <value>London</value>
```

```
  </field>
</SimpleMetadataDocument>
```

### Retrieve all metadata for a subtree

**GET {key-value-metadata-resource}/**(*keypath*)
Retrieves all key-value pairs associated with the specified key path. The key path can contain * for a wildcard segment, or ** for any number of arbitrary segments.

**Produces**

- **application/xml, application/json** – A SimpleMetadataDocument containing all key-value pairs matching.

### Example

```
GET /user/myuser/metadata/**/id
```

### Create multiple key-value pairs

**PUT {key-value-metadata-resource}**
Sets all the specified key-value pairs.

**Accepts**

- **application/xml, application/json** – SimpleMetadataDocument

**PUT {key-value-metadata-resource}/**(*prefix*)
Sets all the specified key-value pairs, prefixed by key path. Key path may not contain wildcards.

**Accepts**

- **application/xml, application/json** – SimpleMetadataDocument

### Example

```
PUT /user/myuser/metadata
Content-Type: application/xml

<SimpleMetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <key>occupation</key>
    <value>developer</value>
  </field>
  <field>
    <key>location</key>
    <value>London</value>
  </field>
</SimpleMetadataDocument>
```

```
200 OK
```

### Delete all key-value pairs

**DELETE {key-value-metadata-resource}**
Clears all key-value pairs for the specified entity.

### Example

```
DELETE /user/myuser/metadata
```

```
200 OK
```

### Retrieve the metadata for a specific key

**GET {key-value-metadata-resource}/**(*keypath*)
Retrieves the value of a specific key. If a key path is specified, exactly one key-value pair must match the key path, else an error is returned.

Key paths can also be specified as well as specific keys.

**Produces**

- **text/plain** – The raw string value.

### Example

```
GET /user/myuser/metadata/location
```

```
London
```

### Set the value for a specific key

**PUT {key-value-metadata-resource}/**(*keypath*)
Sets the value for a specific key. The key path may not contain wildcards.

**Accepts**

- **text/plain** – The raw string value.

### Example

```
PUT /user/myuser/metadata/location
Content-Type: text/plain

Stockholm
```

```
200 OK
```

### Delete key-value pairs

**DELETE {key-value-metadata-resource}/**(*keypath*)
Deletes the key-value pair with the specified key. If a key path is given, it may include wildcards for deleting multiple keys.

Key paths can also be specified as well as specific keys.

### Example

```
DELETE /user/myuser/metadata/location
```

```
200 OK
```

### 17.16.6 Metadata

This page describes metadata related resources for item and collection.

In the following reference, `{metadata-entity}` is one of the following:

- `/item`
- `/collection`

#### Get metadata for an entity

#### Retrieve metadata

**GET** **{metadata-entity}/**(*entity-id*)**/metadata**
Returns the *metadata set* for an entity. This means all metadata change sets, combined, and then filtered according to query parameters.

Conflicts are normally returned with all possible values. With conflict=no, a user interface may choose to receive only one value; i.e., automatic conflict resolution will be enforced. The conflict resolution is only applied to the returned XML document, not to metadata in database.

**Query Parameters**

- **projection** (*string*) – (only supported in item metadata)

  Return metadata set according to projection. Default is `default`.

- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **starttc** (*boolean*) –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.

  - *– group-name* - Exclude specified group.
  - (default) - Return all groups.
- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 – t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **language** (*string*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **conflict** (*string*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **samplerate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **revision** (*string*) – A *change-set-id*, retrieves metadata the way it looked at the given revision.
- **terse** (*string*) – (only supported in item metadata)
  - `yes` - Return metadata in *terse format*
  - `no` (default) - Return metadata in verbose format
- **include** (*string*) – Comma-separated list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **from** (*string*) – (only supported in item metadata)

  A timestamp value. Return metadata starting from the specific timestamp (inclusive)
- **to** (*string*) – (only supported in item metadata)

  A timestamp value. Return metadata until the specific timestamp (non-inclusive)
- **includeConstraintValue** (*string*) – Comma-separated list of fields whose "display value" should be retrieved from the *metadata dataset*.
  - `all` (default) - Return the "display value" of all fields.
  - `none` - No "display value" will be returned. The fields will only have `id` set.
  - *comma-separated field names* - Return the "display value" of the specified fields.

- **includeTransientMetadata** (*boolean*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.

**Produces**

- **application/xml, application/json** –
  - MetadataListDocument for items
  - MetadataDocument for collections
  - or according to specified outgoing projection

**Role** _metadata_read

### Examples

```
GET /item/VX-7888/metadata?field=audio-comments:comment&track=A3&interval=40-60&
↪samplerate=PAL&language=en
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <item id="VX-7888">
        <metadata>
            <timespan start="1000/PAL" end="1250/PAL">
                <field>
                    <name>comment</name>
                    <value lang="en_US" user="joed" site="VY" timestamp="2009-10-
↪11T11:36:30.330+02:00">Music</value>
                </field>
            </timespan>
            <timespan start="1250/PAL" end="1500/PAL">
                <field conflict="yes">
                    <name>comment</name>
                    <value lang="en_US" user="joed" site="VY" timestamp="2009-10-
↪11T11:36:34.527+02:00">Congressman Smith</value>
                    <value lang="en_US" user="bigc" site="VX" timestamp="2009-10-
↪11T11:32:30.330+02:00">Congressman Smythe</value>
                </field>
            </timespan>
        </metadata>
    </item>
</MetadataListDocument>
```

```
GET /item/VX-7888/metadata?track=A3&interval=1000/25-1500/25&conflict=no
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <metadata>
        <item id="VX-7888">
            <timespan start="1000/25" end="1250/25">
                <field>
                    <name>audio-comments</name>
                    <value lang="en_US" user="joed" site="VY" timestamp="2009-10-
↪11T11:36:30.330+02:00">Music</value>
                    <value lang="sv_SE" user="karin" site="VS" timestamp="2009-10-
↪11T14:11:14.888+02:00">Musik</value>
                </field>
            </timespan>
```

```
            <timespan start="1250/25" end="1500/25">
                <field conflict="yes">
                    <name>audio-comments</name>
                    <value lang="en_US" user="joed" site="VY" timestamp="2009-10-
↪11T11:36:34.527+02:00">Congressman Smith</value>
                    <value lang="sv_SE" user="karin" site="VS" timestamp="2009-10-
↪11T14:13:10.100+02:00">Kongressledamot Smith</value>
                </field>
            </timespan>
        </item>
    </metadata>
</MetadataListDocument>
```

## Manipulating change sets

### Create a metadata change set

**PUT {metadata-entity}/**(*entity-id*)**/metadata**

Sets the metadata for an entity, or, more specifically, creates a metadata *change set* for an entity.

The metadata change set binds to different intervals, tracks, and languages, which can be specified either in the URL or in the XML. Providing an empty timespan or an empty field will be interpreted as the removal of any existing element that matches. Fields specified by the system will not be removed by this action.

The revision can either be specified in the input XML/JSON or as a query parameter. If it is not set at all, it will attempt to override any existing values.

**Query Parameters**

- **revision** (*string*) – The known revision. If not specified, the change set will attempt to override existing change sets.
- **skipForbidden** (*boolean*) – Skip fields or groups that the user doesn't have write access to. Default is `false`
- **projection** (*string*) – (only supported in item metadata)

  Sets metadata set according to projection. Default is `default`
- **output-projection** (*string*) – (only supported in item metadata)

  Returns metadata according to projection. Default is `default`
- **onlyReturnChanges** (*boolean*) – New in version 4.16.6.
  - `true` - Only return the changed entries.
  - `false` (default) - Return the whole global metadata after the update.

**Status Codes**

- **400** – Invalid input.
- **404** – Invalid id.

**Accepts**

- **application/json, application/xml** – MetadataDocument or according to specified projection

**Produces**

- **application/xml, application/json** –

- MetadataListDocument for items
- MetadataDocument for collections
- or according to specified outgoing projection

**Role** _metadata_write

### Move metadata

**PUT {metadata-entity}/**(*entity-id*)**/metadata/move**

Moves the specified field or group from one timespan to another. There are some restrictions to this operation:

1. Only top-level elements can be moved, i. e. no groups or fields that belongs to a group can be moved.
2. All conflicts for the specified element must first be resolved before moving it.
3. If moving a field, it cannot be set as sortable.
4. If moving a field, it cannot be system specified.

**Query Parameters**

- **start** (*string*) – The new start *time code*
- **end** (*string*) – The new end *time code*
- **uuid** (*string*) – Required. The UUID of the element.

**Status Codes**

- **400** – Invalid input.
- **404** – Invalid id.

**Produces**

- **application/xml, application/json** – MetadataDocument

**Role** _metadata_write

### Example

Retrieving the current metadata and checking the UUID of the top-level group element.

```
GET /item/VX-7620/metadata
```

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan end="18" start="17">
      <group change="VX-16293" timestamp="2010-09-07T16:41:09.045+02:00" user="admin"␣
↪uuid="96635ac0-1242-496b-ae14-100de8934a2c">
         <name>myfieldgroup</name>
         <group change="VX-16293" timestamp="2010-09-07T16:41:09.045+02:00" user=
↪"admin" uuid="eada6004-7b7e-4000-8707-d6797ed27d72">
            <name>myfieldgroup</name>
            <field change="VX-16293" timestamp="2010-09-07T16:41:09.045+02:00" user=
↪"admin" uuid="03a37ea1-ab96-4c15-af6d-9a0efcac97f0">
               <name>title</name>
               <value change="VX-16293" timestamp="2010-09-07T16:41:09.045+02:00"␣
↪user="admin" uuid="fa205556-f2cc-4456-8e54-075828e9da81">This is my title.</value>
            </field>
         </group>
      </group>
```

```
   </timespan>
</MetadataDocument>
```

Moving the top-level element to the timespan (-INF, +INF):

```
PUT /item/VX-7620/metadata/move?uuid=96635ac0-1242-496b-ae14-100de8934a2c&start=-INF&
 ↪end=%2BINF
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan end="+INF" start="-INF">
      <group change="VX-16293" timestamp="2010-09-07T16:41:09.045+02:00" user="admin"␣
 ↪uuid="96635ac0-1242-496b-ae14-100de8934a2c">
         <name>myfieldgroup</name>
         <group change="VX-16293" timestamp="2010-09-07T16:41:09.045+02:00" user=
 ↪"admin" uuid="eada6004-7b7e-4000-8707-d6797ed27d72">
            <name>myfieldgroup</name>
            <field change="VX-16293" timestamp="2010-09-07T16:41:09.045+02:00" user=
 ↪"admin" uuid="03a37ea1-ab96-4c15-af6d-9a0efcac97f0">
               <name>title</name>
               <value change="VX-16293" timestamp="2010-09-07T16:41:09.045+02:00"␣
 ↪user="admin" uuid="fa205556-f2cc-4456-8e54-075828e9da81">This is my title.</value>
            </field>
         </group>
      </group>
   </timespan>
</MetadataDocument>
```

### List all change sets

**GET {metadata-entity}/**(*entity-id*)**/metadata/changes**
Retrieves change sets that have been applied to the metadata.

**Query Parameters**

- **change** (*string*) – Retrieve a single change set.
- **first** (*integer*) – Return change sets from that number in the list of sorted change sets. Default is `1`.
- **number** (*integer*) – Return at most that number of change sets. Default is `0` (all change sets).
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **starttc** (*boolean*) –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.
- **field** (*string*) – Comma-separated list.

  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 − t2* - Return metadata for specified track interval, e.g. `A2−4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **language** (*string*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **conflict** (*string*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **samplerate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **include** (*string*) – Comma-separated list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **from** (*string*) – (only supported in item metadata)

  A timestamp value. Return metadata starting from the specific timestamp (inclusive)
- **to** (*string*) – (only supported in item metadata)

  A timestamp value. Return metadata until the specific timestamp (non-inclusive)

- **includeConstraintValue** (*string*) – Comma-separated list of fields whose "display value" should be retrieved from the *metadata dataset*.
  - `all` (default) - Return the "display value" of all fields.
  - `none` - No "display value" will be returned. The fields will only have `id` set.
  - *comma-separated field names* - Return the "display value" of the specified fields.
- **includeTransientMetadata** (*boolean*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.

**Status Codes**

- **404** – Could not find the entity.

**Response Headers**

- **Link** – Contains URLs to the previous, next, first and last pages.

**Produces**

- **application/xml, application/json** – MetadataChangeSetDocument

**Role** _metadata_read

### Example

```
GET item/VX-250/metadata/changes
```

```
<MetadataChangeSetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <changeSet>
    <id>VX-30</id>
    <metadata>
      <revision>VX-30</revision>
      <timespan start="-INF" end="+INF">
        <field>
          <name>durationSeconds</name>
          <value user="system" timestamp="2010-03-19T09:08:09.563+01:00" change="VX-30
↪">232.32</value>
        </field>
        <field>
          <name>durationTimeCode</name>
          <value user="system" timestamp="2010-03-19T09:08:09.576+01:00" change="VX-30
↪">232320000@1000000</value>
        </field>
      </timespan>
    </metadata>
  </changeSet>
  <changeSet>
    <id>VX-31</id>
    <metadata>
      <revision>VX-31</revision>
      <timespan start="-INF" end="+INF">
        <field>
          <name>title</name>
          <value user="admin" timestamp="2010-03-19T09:16:25.454+01:00" change="VX-31
↪">u1's title</value>
        </field>
```

```
      </timespan>
    </metadata>
  </changeSet>
  <changeSet>
    <id>VX-32</id>
    <metadata>
      <revision>VX-32</revision>
      <timespan start="-INF" end="+INF">
        <field>
          <name>title</name>
          <value user="admin" timestamp="2010-03-19T09:16:56.419+01:00" change="VX-32
↪">u2's title</value>
        </field>
      </timespan>
    </metadata>
  </changeSet>
  <changeSet>
    <id>VX-33</id>
    <metadata>
      <revision>VX-33</revision>
      <timespan start="-INF" end="+INF">
        <field>
          <name>title</name>
          <value user="admin" timestamp="2010-03-19T09:21:28.692+01:00" change="VX-33
↪">u1's and u2's title</value>
        </field>
      </timespan>
    </metadata>
  </changeSet>
</MetadataChangeSetDocument>
```

### Retrieve a change set

**GET** **{metadata-entity}/**(*entity-id*)**/metadata/changes/***changeset-id*
Retrieves a specific change set.

**Status Codes**

- **404** – Could not find the entity or the change set.

**Produces**

- **application/xml, application/json** – A MetadataChangeSetDocument containing the metadata from the change.

**Role** _metadata_read

### Compare two change sets

**GET** **{metadata-entity}/**(*entity-id*)**/metadata/changes/***changeset-id***/compareTo/***from-changeset-id*
Retrieves a metadata document containing the differences between the entity metadata as of revision `changeset-id` compared to the metadata as of revision `from-changeset-id`.

The `mode` attribute is used to indicate if a field, field group or value has been added or removed.

Note: This should be seen as a diff between the metadata as it was between the two revisions, meaning that for example fields or field groups that have been added and then removed in between will not be shown.

**Parameters**

- **from-changeset-id** (*string*) – The id of the change set to compare against. Use `previous` to compare with the preceding change set.

**Query Parameters**

- **valuesByUuid** (*boolean*) – If `true` (default) then field values will be compared by uuid, if `false` then by the value itself.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 − t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* ⋆ - Return metadata for all tracks of specified type, e.g. `A⋆`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **language** (*string*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `⋆_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **samplerate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.

- **conflict** (*string*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **include** (*string*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.

**Status Codes**

- **404** – Could not find the item.

**Produces**

- **application/xml, application/json** – MetadataDocument

**Role** _metadata_read

### Example

Retrieving the current metadata of the item:

```
GET item/VX-250/metadata/
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item id="VX-12621">
      <metadata>
         <revision>VX-41277,VX-41278</revision>
         <timespan end="+INF" start="-INF">
            <field change="VX-41277" timestamp="2013-04-17T15:20:44.686+02:00" user=
↪"system" uuid="6ddc9537-8c10-46ff-9ed8-5f12bedc589a">
               <name>itemId</name>
               <value change="VX-41277" timestamp="2013-04-17T15:20:44.686+02:00"␣
↪user="system" uuid="d26312bf-e240-4534-9695-670be5c7bb76">VX-12621</value>
            </field>
            <field change="VX-41277" timestamp="2013-04-17T15:20:44.686+02:00" user=
↪"system" uuid="048ebf22-43b7-4979-abe0-8aa5c12363ad">
               <name>created</name>
               <value change="VX-41277" timestamp="2013-04-17T15:20:44.686+02:00"␣
↪user="system" uuid="8eeb6df4-92f4-4d94-96be-ad1b8dd7a06b">2013-04-17T13:20:44.512Z</
↪value>
            </field>
         </timespan>
      </metadata>
   </item>
</MetadataListDocument>
```

Set the item title:

```
PUT /item/VX-250/metadata/
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan start="-INF" end="+INF">
      <field>
```

```
        <name>title</name>
        <value>New title</value>
      </field>
   </timespan>
</MetadataDocument>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item>
      <metadata>
         <revision>VX-41277,VX-41278,VX-41279</revision>
         <timespan end="+INF" start="-INF">
            <field change="VX-41279" timestamp="2013-04-17T15:24:00.907+02:00" user=
↪"admin" uuid="2332c696-b3c8-4a55-9d37-437043258411">
               <name>title</name>
                  <value change="VX-41279" timestamp="2013-04-17T15:24:00.907+02:00"␣
↪user="admin" uuid="dd139c76-86cf-4826-9037-892c928818d9">New title</value>
            </field>
            <field change="VX-41277" timestamp="2013-04-17T15:20:44.686+02:00" user=
↪"system" uuid="6ddc9537-8c10-46ff-9ed8-5f12bedc589a">
               <name>itemId</name>
               <value change="VX-41277" timestamp="2013-04-17T15:20:44.686+02:00"␣
↪user="system" uuid="d26312bf-e240-4534-9695-670be5c7bb76">VX-12621</value>
            </field>
            <field change="VX-41277" timestamp="2013-04-17T15:20:44.686+02:00" user=
↪"system" uuid="048ebf22-43b7-4979-abe0-8aa5c12363ad">
               <name>created</name>
               <value change="VX-41277" timestamp="2013-04-17T15:20:44.686+02:00"␣
↪user="system" uuid="8eeb6df4-92f4-4d94-96be-ad1b8dd7a06b">2013-04-17T13:20:44.512Z</
↪value>
            </field>
         </timespan>
      </metadata>
   </item>
</MetadataListDocument>
```

Retrieving the changes since the last update:

```
GET item/VX-250/metadata/changes/VX-41279/compareTo/previous
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan start="-INF" end="+INF">
      <field uuid="2332c696-b3c8-4a55-9d37-437043258411" user="admin" timestamp="2013-
↪04-17T15:24:00.907+02:00" change="VX-41279" mode="add">
         <name>title</name>
         <value uuid="dd139c76-86cf-4826-9037-892c928818d9">New title</value>
      </field>
   </timespan>
</MetadataDocument>
```

### Update a change set

**PUT {metadata-entity}/**(*entity-id*)**/metadata/changes/**
*changeset-id* Replaces the contents of a change set with the specified id with the metadata given in the document.

**Status Codes**

- **404** – Could not find the item or the change set.

**Accepts**

- **application/xml, application/json** – A MetadataDocument containing the new version of the change set.

**Produces**

- **application/xml, application/json** – A MetadataDocument containing the metadata of the item.

**Role** _metadata_write

### Example

Retrieving the current metadata of the item:

```
GET item/VX-250/metadata/
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item id="VX-250">
      <metadata>
         <revision>VX-15930</revision>
         <timespan end="+INF" start="-INF">
               <name>durationSeconds</name>
               <value change="VX-15930" timestamp="2010-07-02T10:36:20.317+02:00"␣
↪user="system">14.118</value>
            </field>
            <field>
               <name>durationTimeCode</name>
               <value change="VX-15930" timestamp="2010-07-02T10:36:20.317+02:00"␣
↪user="system">14118000@1000000</value>
            </field>
         </timespan>
      </metadata>
   </item>
</MetadataListDocument>
```

Inserting some metadata:

```
PUT /item/VX-250/metadata/
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan end="8" start="5">
      <field>
         <name>title</name>
         <value lang="en_US">My title!</value>
      </field>
      <field>
         <name>my_field</name>
         <value lang="en_US">4</value>
      </field>
   </timespan>
</MetadataDocument>
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item id="VX-250">
```

```
      <metadata>
         <revision>VX-15930,VX-15932</revision>
         <timespan end="+INF" start="-INF">
               <name>durationSeconds</name>
               <value change="VX-15930" timestamp="2010-07-02T10:36:20.317+02:00"␣
↪user="system">14.118</value>
            </field>
            <field>
               <name>durationTimeCode</name>
               <value change="VX-15930" timestamp="2010-07-02T10:36:20.317+02:00"␣
↪user="system">14118000@1000000</value>
            </field>
         </timespan>
         <timespan end="8" start="5">
            <field>
               <name>title</name>
               <value change="VX-15932" lang="en_US" timestamp="2010-07-02T10:39:31.
↪170+02:00" user="admin">My title!</value>
            </field>
            <field>
               <name>my_field</name>
               <value change="VX-15932" lang="en_US" timestamp="2010-07-02T10:39:31.
↪168+02:00" user="admin">4</value>
            </field>
         </timespan>
      </metadata>
   </item>
</MetadataListDocument>
```

Modifying that change set:

```
PUT /item/VX-250/metadata/changes/VX-15932
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan end="15" start="8">
      <field>
         <name>title</name>
         <value lang="en_US">My title!</value>
      </field>
   </timespan>
</MetadataDocument>
```

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <revision>VX-15930,VX-15932</revision>
   <timespan end="+INF" start="-INF">
      <field>
         <name>durationSeconds</name>
         <value change="VX-15930" timestamp="2010-07-02T10:36:20.317+02:00" user=
↪"system">14.118</value>
      </field>
      <field>
         <name>durationTimeCode</name>
         <value change="VX-15930" timestamp="2010-07-02T10:36:20.317+02:00" user=
↪"system">14118000@1000000</value>
      </field>
      </timespan>
      <timespan end="15" start="8">
```

```
        <field>
           <name>title</name>
           <value change="VX-15932" lang="en_US" timestamp="2010-07-02T10:39:31.
↪170+02:00" user="admin">My title!</value>
        </field>
     </timespan>
</MetadataDocument>
```

### Update multiple change sets

**PUT** **{metadata-entity}/**(*entity-id*)**/metadata/changes**
Replaces the metadata in the specified change sets with the given data.

**Statuscode** Could not find the entity or the change set.

**Accepts**

- **application/xml, application/json** – A MetadataChangeSetDocument containing the change sets that should be modified.

**Produces**

- **application/xml, application/json** – A MetadataChangeSetDocument containing a list of the modified change sets.

**Role** _metadata_write

### Trim a change set

**PUT** **{metadata-entity}/**(*entity-id*)**/metadata/changes/**
*changeset-id***/trim** Removes fields and values from the change set(s) that did not result in an actual change of the metadata.

Note that if all fields of a change set are removed, then the change set will also be removed.

**Status Codes**

- **404** – Could not find the item or the change set.

**Produces**

- **application/xml, application/json** – A MetadataChangeSetDocument containing a list of the modified change sets.

**Role** _metadata_write

### Trim multiple change sets

**PUT** **{metadata-entity}/**(*entity-id*)**/metadata/changes/trim**
Removes fields and values from the change set(s) that did not result in an actual change of the metadata.

Note that if all fields of a change set are removed, then the change set will also be removed.

**Status Codes**

- **404** – Could not find the item or the change set.

**Produces**

- **application/xml, application/json** – A MetadataChangeSetDocument containing a list of the modified change sets.

**Role** _metadata_write

### Example

Given an item with multiple change sets for the same set of fields:

```
GET /item/VX-260/metadata/changes
```

```
<MetadataChangeSetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <changeSet>
    <id>VX-816670</id>
    <metadata>
      <revision/>
      <timespan start="-INF" end="+INF">
        <field uuid="0e6bb918-090a-4388-9a55-e039a0462a20" user="admin" timestamp=
↪"2015-02-19T18:24:51.234+01:00" change="VX-816670">
          <name>title</name>
          <value uuid="9b2b24a5-9426-4f10-a37c-6ec0de90b07e" user="admin" timestamp=
↪"2015-02-19T18:24:51.234+01:00" change="VX-816670">A</value>
        </field>
      </timespan>
    </metadata>
  </changeSet>
  <changeSet>
    <id>VX-816671</id>
      <metadata>
        <revision>VX-816669,VX-816670,VX-816668</revision>
        <timespan start="-INF" end="+INF">
          <field uuid="0e6bb918-090a-4388-9a55-e039a0462a20" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">
            <name>title</name>
            <value uuid="b7d9a4ad-a564-4d03-abf5-280f5ad69658" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">B</value>
          </field>
        <field uuid="ea7a2b4a-c105-4cb0-a55b-dc0aa9457e82" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">
          <name>created</name>
          <value uuid="35f3b25c-c4ca-48e7-865b-2918fa1d33e0" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">2015-02-19T17:24:50.262Z</value>
        </field>
        <field uuid="452c038e-b7c8-4076-be40-43210b07c004" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">
          <name>mediaType</name>
          <value uuid="018a0d40-6a0b-4238-b37f-d20dd9720d3a" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">none</value>
        </field>
        <field uuid="57e830d5-e06c-47bd-9495-e0f5d78e0a99" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">
          <name>itemId</name>
          <value uuid="5f1add70-1ff6-445f-a72d-e80e7734d399" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">VX-415819</value>
        </field>
        <field uuid="66efebb4-fbb3-4b7a-8cac-fbb4f68e28e8" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">
          <name>shapeTag</name>
          <value uuid="6402586c-a9dc-45f7-a676-b133fc38e7b3" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">original</value>
        </field>
      </timespan>
    </metadata>
```

```
  </changeSet>
</MetadataChangeSetDocument>
```

Trimming the change sets will then cause fields where the values are unchanged to be removed.

```
PUT /item/VX-260/metadata/changes/trim
```

```
<MetadataChangeSetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <changeSet>
    <id>VX-816670</id>
    <metadata>
      <revision/>
      <timespan start="-INF" end="+INF">
        <field uuid="0e6bb918-090a-4388-9a55-e039a0462a20" user="admin" timestamp=
↪"2015-02-19T18:24:51.234+01:00" change="VX-816670">
          <name>title</name>
          <value uuid="9b2b24a5-9426-4f10-a37c-6ec0de90b07e" user="admin" timestamp=
↪"2015-02-19T18:24:51.234+01:00" change="VX-816670">A</value>
        </field>
      </timespan>
    </metadata>
  </changeSet>
  <changeSet>
    <id>VX-816671</id>
    <metadata>
      <revision>VX-816670</revision>
      <timespan start="-INF" end="+INF">
        <field uuid="0e6bb918-090a-4388-9a55-e039a0462a20" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">
          <name>title</name>
          <value uuid="b7d9a4ad-a564-4d03-abf5-280f5ad69658" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">B</value>
        </field>
      </timespan>
    </metadata>
  </changeSet>
```

### Delete a change set

**DELETE {metadata-entity}/** (*entity-id*) **/metadata/changes/**
*changeset-id* Deletes an entire change set.

**Status Codes**

- **404** – Could not find the entity or the change set.

**Produces**

- **application/xml, application/json** – A MetadataDocument containing the metadata of the item.

**Role** _metadata_write

### Example

Retrieving the current metadata:

```
GET /item/VX-250/metadata
```

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <item id="VX-250">
      <metadata>
         <revision>VX-15930,VX-15932</revision>
         <timespan end="+INF" start="-INF">
               <name>durationSeconds</name>
               <value change="VX-15930" timestamp="2010-07-02T10:36:20.317+02:00"␣
↪user="system">14.118</value>
            </field>
            <field>
               <name>durationTimeCode</name>
               <value change="VX-15930" timestamp="2010-07-02T10:36:20.317+02:00"␣
↪user="system">14118000@1000000</value>
            </field>
         </timespan>
         <timespan end="8" start="5">
            <field>
               <name>title</name>
               <value change="VX-15932" lang="en_US" timestamp="2010-07-02T10:39:31.
↪170+02:00" user="admin">My title!</value>
            </field>
            <field>
               <name>my_field</name>
               <value change="VX-15932" lang="en_US" timestamp="2010-07-02T10:39:31.
↪168+02:00" user="admin">4</value>
            </field>
         </timespan>
      </metadata>
   </item>
</MetadataListDocument>
```

Deleting the change set “VX-15932”:

```
DELETE /item/VX-250/metadata/changes/VX-15932
```

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <revision>VX-15930</revision>
   <timespan end="+INF" start="-INF">
      <field>
         <name>durationSeconds</name>
         <value change="VX-15930" timestamp="2010-07-02T10:36:20.317+02:00" user=
↪"system">14.118</value>
      </field>
      <field>
         <name>durationTimeCode</name>
         <value change="VX-15930" timestamp="2010-07-02T10:36:20.317+02:00" user=
↪"system">14118000@1000000</value>
      </field>
      </timespan>
</MetadataDocument>
```

### Modifying metadata using metadata entries

Metadata can also be modified using a MetadataEntryDocument or a MetadataEntryListDocument. They reference existing fields/groups/values with UUID.

### Modify a metadata entry

**PUT {metadata-entity}/**(*entity-id*)**/metadata/entry/**
*entry-uuid* Modifies a specific metadata field/group/value by UUID.

**Accepts**

- **application/xml, application/json** – MetadataEntryDocument

**Produces**

- **application/xml, application/json** – MetadataDocument

**Role** _metadata_write

### Example

Assume we have item VX-10 with the following metadata:

```
<MetadataListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <item id="VX-10">
    <metadata>
      <revision>VX-110,VX-109,VX-148</revision>
      <timespan end="+INF" start="-INF">
        <field change="VX-109" timestamp="2013-11-12T09:39:54.767+01:00" user="system
↪" uuid="31514647-8cf8-4d74-b674-e5941e007e77">
          <name>created</name>
          <value change="VX-109" timestamp="2013-11-12T09:39:54.767+01:00" user=
↪"system" uuid="afeb8aae-093f-4679-a156-eccf6aa5ba63">2013-11-12T08:39:54.670Z</
↪value>
        </field>
        <field change="VX-109" timestamp="2013-11-12T09:39:54.767+01:00" user="system
↪" uuid="5ba7c8f1-d77b-45fb-a8ed-ceda1d794207">
          <name>itemId</name>
          <value change="VX-109" timestamp="2013-11-12T09:39:54.767+01:00" user=
↪"system" uuid="01934732-9c66-4c6d-90a1-f14b8d188612">VX-10</value>
        </field>
        <field change="VX-148" timestamp="2013-11-13T12:13:12.160+01:00" user="admin"␣
↪uuid="fd59a9f0-fe18-4183-82cb-e11aa33ad4bd">
          <name>title</name>
          <value change="VX-148" timestamp="2013-11-13T12:13:12.160+01:00" user="admin
↪" uuid="17d6bac2-56c4-4117-b0dd-da474912bc3c">my item's title</value>
        </field>
        <field change="VX-148" timestamp="2013-11-13T12:13:12.160+01:00" user="admin"␣
↪uuid="e2e964d4-5376-47f0-83dd-f4e3663181d8">
          <name>item_tags</name>
          <value change="VX-148" timestamp="2013-11-13T12:13:12.160+01:00" user="admin
↪" uuid="45588018-2c14-4627-a889-b71559f1a318">tag 1</value>
          <value change="VX-148" timestamp="2013-11-13T12:13:12.160+01:00" user="admin
↪" uuid="a1054ba4-da2d-46bf-8a87-e11bda2df243">tag 2</value>
        </field>
      </timespan>
    </metadata>
  </item>
</MetadataListDocument>
```

And we want to change the title of this item's metadata. Note that the UUID of the title value is `17d6bac2-56c4-4117-b0dd-da474912bc3c`. We can then make the following request;

```
PUT /item/VX-10/metadata/entry/17d6bac2-56c4-4117-b0dd-da474912bc3c
Content-Type: application/xml

<MetadataEntryDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <value>my item's new title</value>
</MetadataEntryDocument>
```

If we instead want to replace the values in the `item_tags` field, we can make the following request:

```
PUT /item/VX-10/metadata/entry/e2e964d4-5376-47f0-83dd-f4e3663181d8
Content-Type: application/xml

<MetadataEntryDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>item_tags</name>
    <value>new tag 1</value>
    <value>new tag 2</value>
  </field>
</MetadataEntryDocument>
```

This works in a similar fashion for field groups too.

### Modify multiple metadata entries

**PUT {metadata-entity}/** (*entity-id*) **/metadata/entry**
Modifies multiple metadata fields/groups/values by UUID.

**Accepts**

- **application/xml, application/json** – MetadataEntryListDocument

**Produces**

- **application/xml, application/json** – MetadataDocument

**Role** _metadata_write

### Example

The above changes could be made in one request in the following way:

```
PUT /item/VX-10/metadata/entry
Content-Type: application/xml

<MetadataEntryListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <entry uuid="17d6bac2-56c4-4117-b0dd-da474912bc3c">
    <value>my item's new title</value>
  </entry>
  <entry uuid="e2e964d4-5376-47f0-83dd-f4e3663181d8">
    <field>
      <name>item_tags</name>
      <value>new tag 1</value>
      <value>new tag 2</value>
    </field>
  </entry>
</MetadataEntryListDocument>
```

#### Metadata visualization

#### Retrieve the metadata graph

**GET {metadata-entity}/**(*entity-id*)**/metadata/graph**
Shows fields and values and their history as a graph.

**Query Parameters**

- **groupBy**(*string*) –
  - `change` (default) - Group fields and values by change set.
  - `none` - Present fields and groups without any grouping.

**Produces**

- **image/png** –

**Role** _administrator

#### Retrieve the metadata graph as dot file

**GET {metadata-entity}/**(*entity-id*)**/metadata/graph/dot**
Shows fields and values and their history in dot format, for further processing.

**Query Parameters**

- **groupBy**(*string*) –
  - `change` (default) - Group fields and values by change set.
  - `none` - Present fields and groups without any grouping.

**Produces**

- **text/plain, text/vnd.graphviz** –

**Role** _administrator

### 17.16.7 Re-indexing metadata

The text index can be fully reindexed without disturbing production. The reindexing process also handles application server restarts, and can be monitored. Normally, reindexing is only needed if the metadata field definition are changed, but it can be a good idea to reindex if the database is moved or if upgrading to a newer version of Vidispine.

#### Controlling index of items/collections

If the metadata field `solr-index` of an item/collection is set to `false`, it won't be indexed.

Example:

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan end="+INF" start="-INF">
      <field>
         <name>solr-index</name>
         <value>false</value>
      </field>
   </timespan>
</MetadataDocument>
```

**Note:** The number of indexes returned in the `GET` command is only an estimate.

Changed in version 5.0: The `document` index was added.

### Request a reindex

**`PUT /reindex/`** (*index*)
Starts a reindex. If a reindex of the same type is already in progress, it is restarted.

The types than can be reindexed are: item, collection, ACL, file, thumbnail and document.

**Note:** Only files that do *not* belong to an item will be indexed when reindexing files. To reindex all files both a file and item reindex must be started.

**Parameters**

- **`index`** (*`string`*) – The index to rebuild. One of `item`, `collection`, `acl`, `file`, `thumbnail` or `document`.

**Query Parameters**

- **`status`** (*`string`*) – If current status is `FINISHED` or `ABORTED` then `PAUSED`, `ABORTED` and `IN_PROGRESS` are no-ops.
  - `IN_QUEUE` (default) - To start or restart reindex.
  - `PAUSED` - To pause reindex.
  - `ABORTED` - To cancel reindex.
  - `IN_PROGRESS`/`IN PROGRESS` - To resume a paused reindex.
- **`priority`** (*`integer`*) – The priority of this reindex request compare to other request. Requests with a larger/higher priority will be processed first. Default is `500`.

**Produces**

- **`application/xml, application/json`** – ReindexRequestDocument

**Role** _administrator

Example:

```
PUT /reindex/item
```

```
<ReindexRequestDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <index>item</index>
    <priority>500</priority>
    <status>IN QUEUE</status>
    <start>2016-03-23T22:41:04.133+01:00</start>
</ReindexRequestDocument>
```

### Retrieve reindex status

**`GET /reindex/`** (*index*)
Gets information about a reindex process, i.e., progress and whether it is finished.

**Produces**

- **application/xml, application/json** – ReindexRequestDocument

**Role** _administrator

Example:

```
GET /reindex/item
```

```
<ReindexRequestDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <index>item</index>
    <priority>500</priority>
    <status>FINISHED</status>
    <start>2016-03-23T22:41:04.133+01:00</start>
    <finish>2016-03-23T22:41:04.774+01:00</finish>
    <indexesDone>48</indexesDone>
    <indexesTotal>48</indexesTotal>
</ReindexRequestDocument>
```

### 17.16.8 Metadata locks

New in version 4.17: Collection metadata support

A locking container is a stateful resource that holds locks for any number of fields of the metadata of an item or collection.

Operations on metadata locks uses `{metadata-resource}` which is either of

- `/collection`
- `/item`

#### List all locking containers

**GET {metadata-resource}/**(*id*)**/metadata-lock**
Returns all locking containers.

**Produces**

- **application/xml, application/json** – MetadataLockListDocument
- **text/plain** – CRLF-delimited list of locking ids

**Role** _metadata_lock_read

#### Retrieve a locking container

**GET {metadata-resource}/**(*id*)**/metadata-lock/**
*lock-id* Returns information about specified locking container.

**Produces**

- **application/xml, application/json** – MetadataLockDocument

**Role** _metadata_lock_read

#### Create a locking container

**POST {metadata-resource}/**(*id*)**/metadata-lock**
Creates a new locking container, optionally with initial locks.

**Query Parameters**

- **field** (*string*) – Comma-separated list of fields to lock.

- **timeout** (*integer*) – Time-out in seconds. Default is `60`.

**Produces**

- **application/xml, application/json** – MetadataLockDocument
- **text/plain** – Locking id

**Role** _metadata_lock_write

#### Update a locking container

**PUT {metadata-resource}/**(*id*)**/metadata-lock/**
*lock-id* Add new fields to the locking container and/or updates the expiry time.

**Query Parameters**

- **field** (*string*) – Comma-separated list of fields to lock.
- **timeout** (*integer*) – Time-out in seconds. Default is `60`.

**Produces**

- **application/xml, application/json** – MetadataLockDocument
- **text/plain** – Locking id

**Role** _metadata_lock_write

#### Delete a locking container

**DELETE {metadata-resource}/**(*id*)**/metadata-lock/**
*lock-id* Remove the locking container and all locks associated with it.

**Role** _metadata_lock_write

### 17.16.9 Metadata fields

A metadata field is an ingredient of definition of the metadata set. Metadata fields define name and *type* of fields. Metadata fields can be organized into *groups of fields*. Furthermore fields can also be assigned *additional data*.

Access to fields can be restricted using *access controls*.

#### Managing metadata fields

#### List all metadata fields

**GET /metadata-field**
Returns a list of all defined *fields*.

**Query Parameters**

- **includeValues** (*boolean*) – Return the value enumeration for each field. Default is `false`.
- **data** (*string*) – Comma-separated list of any additional data to include.

**Produces**

- **application/xml, application/json** – MetadataFieldListDocument
- **text/plain** – CRLF-delimited list of ids or URLs

**Role** _metadata_field_read

### Retrieve a metadata field

**GET /metadata-field/** (*field-name*)
Returns information about a specific metadata field definition.

**Query Parameters**

- **includeValues** (*boolean*) – Return the value enumeration for this field. Default is `false`.
- **data** (*string*) – Comma-separated list of any additional data to include. Wildcard characters. e.g. `*`, `myapp_*`, `*_version` can be used. (New in 4.17.)

**Status Codes**

- **404 Not found** – The specified field could not be found.

**Produces**

- **application/xml, application/json** – MetadataFieldDocument

**Role** _metadata_field_read

### Update or create a metadata field

**PUT /metadata-field/** (*field-name*)
Updates or creates a metadata field definition.

**Status Codes**

- **400 Bad request** – Either the MetadataFieldDocument was not specified correctly or an illegal type was given.

**Accepts**

- **application/xml, application/json** – MetadataFieldDocument

**Produces**

- **application/xml, application/json** – MetadataFieldDocument

**Role** _metadata_field_write

### Delete a metadata field

**DELETE /metadata-field/** (*field-name*)
Removes the metadata field definition. Note that this action may invalidate existing metadata.

**Status Codes**

- **200 OK** – The field was deleted successfully.
- **404 Not found** – The field could not be found.

**Role** _metadata_field_write

### Field value enumerations and validation

It's possible to get a list of allowed values based on the defined *metadata dataset* on the field.

### Retrieve the enumeration for a field

**GET /metadata-field/** (*field-name*) **/allowed-values**

Returns the allowed values for a field based on the defined metadata dataset on the field.

**Query Parameters**

- **constraint** (*string[]*) – Multiple query parameters can be specified. If no query parameters are specified, all the allowed values are returned.
  - *field-name* = *value* - Apply additional constraint to the resulting value.

    Note that = is part of the query parameter, and has to be encoded (`%3d`).

**Produces**

- **application/xml, application/json** – ConstraintValueListDocument

Example:

```
GET /metadata-field/rdf_city/allowed-values
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ConstraintValueListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <value id="testNS:manchester1">Manchester</value>
  <value id="testNS:nyc1">New York City</value>
  <value id="vidi:gid7">Buffalo</value>
  <value id="vidi:gid1">Los Angeles</value>
  <value id="vidi:gid6">Rochester</value>
  <value id="testNS:london1">London</value>
  <value id="vidi:gid8">San Francisco</value>
</ConstraintValueListDocument>
```

```
GET /metadata-field/rdf_city/allowed-values?constraint=rdf_country=UK
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ConstraintValueListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <value id="testNS:manchester1">Manchester</value>
  <value id="testNS:london1">London</value>
</ConstraintValueListDocument>
```

### Retrieve the enumeration for field given some constraints

**POST /metadata-field/** (*field-name*) **/allowed-values**

Returns the allowed values for a field based on the defined metadata dataset on the field.

**Accepts**

- **application/xml, application/json** – MetadataFieldValueConstraintListDocument

**Produces**

- **application/xml, application/json** – ConstraintValueListDocument

Example request:

```
POST /metadata-field/rdf_city/allowed-values
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MetadataFieldValueConstraintListDocument xmlns="http://xml.vidispine.com/schema/
↪vidispine">
  <constraint>
    <field>rdf_country</field>
    <value>USA</value>
  </constraint>
  <constraint>
    <field>rdf_state</field>
    <value>New York</value>
  </constraint>
</MetadataFieldValueConstraintListDocument>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ConstraintValueListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <value id="testNS:nyc1">New York City</value>
  <value id="vidi:gid7">Buffalo</value>
  <value id="vidi:gid6">Rochester</value>
</ConstraintValueListDocument>
```

### Field value enumerations

Certain fields may have a restricted set of possible values, that a user should be able to select from for example. These value enumerations for a field can be stored in Vidispine.

Enumerations are made up out of a key and a value. The key is what should be stored in the metadata, while the value is the display name or description of the value.

### Retrieve the enumeration for a field

**GET /metadata-field/** (*field-name*) **/values**
Retrieves the value enumeration for a specific field.

•Only one of key-exact, key-start or key-regex can be set.

•Only one of value-exact, value-start or value-regex can be set.

**Query Parameters**

- **key-exact** (*string*) – Return the key with this name.
- **key-start** (*string*) – Return keys starting with this prefix.
- **key-regex** (*string*) – Return keys matching this regular expression.
- **value-exact** (*string*) – Return keys with this value.
- **value-start** (*string*) – Return keys with a value starting with this prefix.
- **value-regex** (*string*) – Return keys with a value matching this regular expression.
- **hits** (*integer*) – The maximum number of keys to return. Default is all.

**Produces**

- **application/xml, application/json** – SimpleMetadataDocument

### Example

```
GET /metadata-field/document_language/values
```

```
<SimpleMetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <key>en</key>
    <value>English</value>
  </field>
  <field>
    <key>sv</key>
    <value>Swedish</value>
  </field>
</SimpleMetadataDocument>
```

### Set the enumeration for a field

**PUT /metadata-field/** (*field-name*) **/values**
Sets the value enumeration for a specific field.

**Accepts**

- **application/xml, application/json** – SimpleMetadataDocument

### Example

```
PUT /metadata-field/document_language/values
```

```
<SimpleMetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <key>en</key>
    <value>English</value>
  </field>
  <field>
    <key>sv</key>
    <value>Swedish</value>
  </field>
</SimpleMetadataDocument>
```

```
200 OK
```

### Field metadata

Metadata fields can be assigned additional data in a key-value fashion. This data can later be seen when retrieving metadata, using the include parameter in *Retrieve metadata*.

Deprecated since version 4.1.2: Use the *Key-value metadata* interface instead.

### Terse metadata schema

### Retrieve terse metadata schema

**GET /metadata-field/terse-schema**
Retrieves the schema that defines terse *metadata*. This schema is dynamically generated based on the fields present in the system.

**Produces**

- **application/xml** – An XML schema.

**Role** _metadata_field_read

### Viewing effective permissions

The merged access resource on fields and field groups can be used to check if a specific user is allowed to view, edit or delete specific fields or groups.

### Retrieve user access to all fields

**GET /metadata-field/merged-access/**
Retrieves the permission for a specific user to all fields.

**Query Parameters**

- **username** (*string*) – The name of the user to check.

**Produces**

- **application/xml, application/json** – An AccessControlMergedDocument containing the fields, user and permissions.

**Role** _accesscontrol_read

### Example

```
GET /metadata-field/merged-access
```

```
<AccessControlMergedDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field username="admin">
    <name>collectionId</name>
    <permission>DELETE</permission>
  </field>
  <field username="admin">
    <name>itemId</name>
    <permission>DELETE</permission>
  </field>
  <field username="admin">
    <name>mediaType</name>
    <permission>DELETE</permission>
  </field>
  <field username="admin">
    <name>mrk_color</name>
    <permission>DELETE</permission>
  </field>
  ...
</AccessControlMergedDocument>
```

### Retrieve user access to field

**GET /metadata-field/** (*field-name*) **/merged-access/**
Retrieves the permission for a specific user to a field.

**Query Parameters**

- **username** (*string*) – The name of the user to check.

**Produces**

- **application/xml, application/json** – An AccessControlMergedDocument containing the field, user and permission.

**Role** _accesscontrol_read

### Example

For a field without any access controls:

```
GET /metadata-field/title/access
```

```
<MetadataFieldAccessControlListDocument xmlns="http://xml.vidispine.com/schema/
↪vidispine"/>
```

We can see that the default, which is to allow full access to fields, is in effect:

```
GET /metadata-field/title/merged-access
```

```
<AccessControlMergedDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field username="admin">
    <name>title</name>
    <permission>DELETE</permission>
  </field>
</AccessControlMergedDocument>
```

### Retrieve user access to all field groups

**GET /metadata-field/field-group/merged-access/**

Retrieves the permission for a specific user to all field groups and the field groups and fields part of those groups.

**Query Parameters**

- **username** (*string*) – The name of the user to check.

**Produces**

- **application/xml, application/json** – An AccessControlMergedDocument containing the group, its children, user and permission.

**Role** _accesscontrol_read

### Example

```
GET /metadata-field/field-group/merged-access
```

```
<AccessControlMergedDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <fieldGroup username="admin">
    <name>facedetect_face</name>
    <permission>DELETE</permission>
    <field username="admin">
      <name>facedetect_pid</name>
      <permission>DELETE</permission>
    </field>
  </fieldGroup>
  ...
</AccessControlMergedDocument>
```

#### Retrieve user access to field group

**GET /metadata-field/field-group/** (*group-name*) **/merged-access/**
Retrieves the permission for a specific user to a field group and the field groups and fields part of that group.

**Query Parameters**

- **username** (*string*) – The name of the user to check.

**Produces**

- **application/xml, application/json** – An AccessControlMergedDocument containing the group, its children, user and permission.

**Role** _accesscontrol_read

#### Example

```
GET /metadata-field/field-group/mrk_marker/merged-access?username=test-user
```

```
<AccessControlMergedDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <fieldGroup username="test-user">
    <name>mrk_marker</name>
    <permission>DELETE</permission>
    <field username="test-user">
      <name>mrk_color</name>
      <permission>DELETE</permission>
    </field>
    <field username="test-user">
      <name>mrk_text</name>
      <permission>DELETE</permission>
    </field>
    <field username="test-user">
      <name>mrk_zerolength</name>
      <permission>DELETE</permission>
    </field>
  </fieldGroup>
</AccessControlMergedDocument>
```

### 17.16.10 Metadata field access controls

Field access controls can be used to control who is able to view and edit certain metadata fields in the metadata of items and collections.

In the following reference, `{field-access-resource}` is one of the following:

- `/metadata-field/{field-name}/access`
- `/metadata-field/field-group/{group-name}/access`

#### Managing metadata field access controls

#### Retrieve an access control list

**GET {field-access-resource}**
Returns the access control list that is applied to the specified field or group.

**Produces**

- **application/xml, application/json** – MetadataFieldAccessControlListDocument

**Role** _administrator

### Example

```
GET /metadata-field/title/access
```

```
<MetadataFieldAccessControlListDocument xmlns="http://xml.vidispine.com/schema/
↪vidispine">
  <access>
    <id>VX-10</id>
    <field>title</field>
    <group>mygroup</group>
    <permission>READ</permission>
  </access>
  <access>
    <id>VX-5</id>
    <field>title</field>
    <user>myuser</user>
    <permission>DELETE</permission>
  </access>
</MetadataFieldAccessControlListDocument>
```

### Create an access control entry

**POST {field-access-resource}**

Creates an entry in the access control list and returns the created entry together with its id.

**Accepts**

- **application/xml, application/json** – MetadataFieldAccessControlDocument

**Produces**

- **application/xml, application/json** – MetadataFieldAccessControlDocument

**Role** _administrator

### Examples

```
POST /metadata-field/title/access
Content-Type: application/xml

<MetadataFieldAccessControlDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <user>admin</user>
    <permission>DELETE</permission>
</MetadataFieldAccessControlDocument>
```

```
<MetadataFieldAccessControlDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <id>VX-11</id>
    <user>admin</user>
    <permission>DELETE</permission>
</MetadataFieldAccessControlDocument>
```

#### Delete an access control entry

**DELETE {field-access-resource}/**(*access-id*)
Removes the access control entry with the specified id.

**Role** _administrator

#### Examples

```
DELETE /metadata-field/title/access/VX-11
```

```
200 OK
```

### 17.16.11 Metadata field groups

Field groups are named sets of *fields* and groups. The structure of groups is defined using a *metadata schema*.

Access to field groups can be restricted using *access controls*.

#### Managing field groups

#### List all field groups

**GET /metadata-field/field-group**
Retrieves all metadata field groups known by the system.

**Query Parameters**

- **content** (*boolean*) –
  - `true` - Return the groups and their members.
  - `false` (default) - Return the group names only.
- **traverse** (*boolean*) –
  - `true` - Traverse any sub-groups in order to retrieve the entire hierarchy.
  - `false` (default) - Only retrieves the names of the members.
- **data** (*string*) – Comma-separated list of any additional data to include.
  - accepts wildcard characters. e.g. `*`, `myapp_*`, `*_version`.

    (New in 4.17.)

**Produces**

- **application/xml, application/json** – MetadataFieldGroupListDocument
- **text/plain** – CRLF-delimited list of field group names.

**Role** _metadata_field_group_read

#### Create an empty field group

**PUT /metadata-field/field-group/**(*group-name*)
Creates a new group with the given name. If a group with that name already exists, this operation does nothing.

**Status Codes**

- **200 OK** – Group created successfully.

**Role** _metadata_field_group_write

### Update or create a field group

**PUT /metadata-field/field-group/** (*group-name*)
Creates a new group with the given name, if it does not already exists, and adds any specified fields and access control entries to it. If the fields does not exist, they will be created. Furthermore any additional data for the fields will be set as well.

**Query Parameters**

- **clear** (*boolean*) –
  - `true` - If the group already exists, clear all content from it before updating.
  - `false` (default) - If the group already exists, append child groups/fields from the input document.

**Status Codes**

- **200 OK** – Group created successfully.

**Accepts**

- **application/xml, application/json** – MetadataFieldGroupDocument

**Role** _metadata_field_group_write

### Example

```
PUT /metadata-field/field-group/myfieldgroup
Content-Type: application/xml

<MetadataFieldGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <data>
    <key>myextradata</key>
    <value>Extradata for the group</value>
  </data>
  <field>
    <name>title</name>
    <data>
      <key>text</key>
      <value>Here is some text.</value>
    </data>
  </field>
  <field>
    <name>durationSeconds</name>
  </field>
  <field>
    <name>this_field_does_not_exist_yet</name>
    <type>string</type>
    <data>
      <key>myextradata</key>
      <value>Some additional data</value>
    </data>
  </field>
  <access>
    <user>admin</user>
    <permission>DELETE</permission>
  </access>
</MetadataFieldGroupDocument>
```

```
200 OK
```

### Delete a field group

**DELETE /metadata-field/field-group/** (*group-name*)
Deletes the group with the given name.

**Status Codes**

- **200 OK** – Group deleted successfully.
- **404 Not found** – No group with that name exists.

**Role** _metadata_field_group_write

### Group contents

### List all fields of a group

**GET /metadata-field/field-group/** (*group-name*)
Retrieves the specified field group.

**Query Parameters**

- **includeValues** (*boolean*) – Return the value enumeration for each field. Default is `false`
- **traverse** (*boolean*) –
  - `true` - Traverse any sub-groups in order to retrieve the entire hierarchy.
  - `false` (default) - Only retrieves the names of the members.
- **data** (*string*) – Comma-separated list of any additional data to include.
  - accepts wildcard characters. e.g. `*`, `myapp_*`, `*_version`.

    (New in 4.17.)

**Produces**

- **application/xml, application/json** – MetadataFieldGroupDocument

**Role** _metadata_field_group_read

### Add a field to a group

**PUT /metadata-field/field-group/** (*group-name*) **/**
*field-name* Adds the field with the specified name to the group. If the field is already contained within the group this operation does nothing.

**Status Codes**

- **200 OK** – Field added successfully.

**Role** _metadata_field_group_write

### Remove a field from a group

**DELETE /metadata-field/field-group/** (*group-name*) **/**
*field-name* Removes the field with the specified name from the group.

**Status Codes**

- **200 OK** – Field removed successfully.
- **404 Not found** – No field with that name is contained within the group.

**Role** _metadata_field_group_write

### Add a group to a group

**PUT /metadata-field/field-group/** (*parent-group-name*) **/group/**
*child-group-name* Adds the group with the specified name to the group. If the group is already contained within the group this operation does nothing.

**Status Codes**

- **200 OK** – Group added successfully.

**Role** _metadata_field_group_write

### Remove a group from a group

**DELETE /metadata-field/field-group/** (*parent-group-name*) **/group/**
*child-group-name* Removes the group with the specified name from the group.

**Status Codes**

- **200 OK** – Group removed successfully.

**Role** _metadata_field_group_write

### Searching for field groups

### Search for groups used in metadata

**PUT /metadata-field/field-group**
Much like *searching for items*, specific fields can be used when searching. The result is a list of used metadata groups that matches the query. Optionally the definition of the group and the value of the group can be retrieved.

---

**Note:** Results will only be returned if *field group indexing* has not been disabled.

---

**Query Parameters**

- **first** (*integer*) – From the resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** (*integer*) – The number of entities to fetch. Default is `100`.
- **traverse** (*boolean*) –
  - `true` - Traverse any sub-groups in order to retrieve the entire hierarchy.
  - `false` (default) - Only retrieves the names of the members.
- **data** (*string*) – Comma-separated list of any additional data to include.

  - accepts wildcard characters. e.g. `*`, `myapp_*`, `*_version`.

    (New in 4.17.)

- **group** (*string*) – Comma-separated list of group names to restrict search in.
- **includeValue** (*boolean*) –
  - `true` - The value is included in the result. I.e., how the group is specified in the metadata.
  - `false` (default) - The value is not included.
- **includeDefinition** (*boolean*) –
  - `true` - The definition of the group is included in the result.
  - `false` (default) - The definition is not included.
- **includeSource** (*boolean*) –
  - `true` - Information about which entity that contains the matching metadata group is included in the result.
  - `false` (default) - Entity information is not included.
- **source** (*string*) –
  - `item` - Only search for groups in item metadata.
  - `collection` - Only search for groups in collection metadata.
  - `global` - Only search for groups in the global metadata.
  - `document` - Only search for groups in the document metadata.

**Accepts**

- **application/xml, application/json** – MetadataFieldGroupSearchDocument

**Produces**

- **application/xml, application/json** – MetadataFieldResultDocument

**Role** _item_search

### Example

Searching for employees named Andrew (please note that the parameter group is set to employee to only search for employees named Andrew):

```
PUT /metadata-field/field-group?includeValue=true&includeDefinition=true&
↪traverse=false&group=employee&includeSource=true
Content-Type: application/xml

<MetadataFieldGroupSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>example_name</name>
    <value>Andrew</value>
  </field>
</MetadataFieldGroupSearchDocument>
```

```
<MetadataFieldResultDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <hits>1</hits>
  <group name="employee" uuid="db26c542-3507-4d3c-a772-55d4627b3d59" start="-INF" end=
↪"+INF">
```

```
    <value uuid="db26c542-3507-4d3c-a772-55d4627b3d59" user="admin" timestamp="2011-
↪01-10T12:30:01.483+01:00" change="VX-11">
      <name>employee</name>
      <field uuid="82738de5-8ce3-4f28-badc-c97924bb5837" user="admin" timestamp="2011-
↪01-10T12:30:01.483+01:00" change="VX-11">
        <name>example_title</name>
        <value uuid="552f5d06-50d5-4109-9017-8b8ce7d101ff" user="admin" timestamp=
↪"2011-01-10T12:30:01.483+01:00" change="VX-11">Editor</value>
      </field>
      <field uuid="9ff7fced-4500-4d11-a8e9-eb8821b42cbc" user="admin" timestamp="2011-
↪01-10T12:30:01.483+01:00" change="VX-11">
        <name>example_name</name>
        <value uuid="45379c11-b30a-4b6c-a93a-eb5229a61905" user="admin" timestamp=
↪"2011-01-10T12:30:01.483+01:00" change="VX-11">Andrew</value>
      </field>
    </value>
    <definition>
      <name>employee</name>
      <schema min="0" max="-1" name="employee"/>
      <field sortable="false">
        <name>example_title</name>
        <schema reference="false" min="0" max="1" name="example_title"/>
        <type>string</type>
      </field>
      <field sortable="false">
        <name>example_name</name>
        <schema reference="false" min="1" max="1" name="example_name"/>
        <type>string</type>
      </field>
    </definition>
    <source>
      <id>VX-15</id>
      <type>item</type>
    </source>
  </group>
</MetadataFieldResultDocument>
```

## 17.16.12 Metadata datasets

Metadata *datasets* can be linked to metadata fields to restrict the *allowed values* for that field.

### Managing datasets

### List all datasets

**GET /metadata/dataset**
Retrieves the list of metadata datasets.

**Produces**

- **application/xml, application/json** – MetadataDatasetListDocument

**Role** _metadata_dataset_read

### Example

```
GET /metadata/dataset
```

```
<?xml version="1.0"?>
<MetadataDatasetListDocument>
  <dataset>
    <name>testmodel1</name>
    <uri>http://localhost:8080/API/metadata/dataset/testmodel1</uri>
  </dataset>
  <dataset>
    <name>testmodel2</name>
    <uri>http://localhost:8080/API/metadata/dataset/testmodel2</uri>
  </dataset>
</MetadataModelListDocument>
```

### Retrieve a dataset

**GET /metadata/dataset/** (*name*)
Retrieves the metadata dataset with the specified name. The returned serialization format of the RDF model is `RDF/XML` or `TURTLE` depending on the request header.

**Produces**

- **application/rdf+xml, text/turtle, application/ld+json** – The RDF model.

**Role** _metadata_dataset_read

### Example

```
GET /metadata/dataset/testmodel1

Accept: text/turtle
```

```
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .
@prefix vidi:  <http://rdf.vidispine.com/id/#> .

vidi:gid2  skos:definition  "country" ;
        skos:member      vidi:gid3 ;
        skos:prefLabel   "United Kingdom" .

vidi:gid1  skos:definition  "city" ;
        skos:prefLabel   "London" .

vidi:gid0  skos:definition  "city" ;
        skos:prefLabel   "Manchester" .

vidi:gid3  a    rdf:Bag ;
        rdf:_1  vidi:gid1 ;
        rdf:_2  vidi:gid0 .
```

```
GET /metadata/dataset/testmodel1

Accept: application/rdf+xml
```

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:skos="http://www.w3.org/2004/02/skos/core#"
```

```
    xmlns:vidi="http://rdf.vidispine.com/id/#">
  <rdf:Description rdf:about="http://rdf.vidispine.com/id/#gid2">
    <skos:prefLabel>United Kingdom</skos:prefLabel>
    <skos:member>
      <rdf:Bag rdf:about="http://rdf.vidispine.com/id/#gid3">
        <rdf:li>
          <rdf:Description rdf:about="http://rdf.vidispine.com/id/#gid1">
            <skos:prefLabel>London</skos:prefLabel>
            <skos:definition>city</skos:definition>
          </rdf:Description>
        </rdf:li>
        <rdf:li>
          <rdf:Description rdf:about="http://rdf.vidispine.com/id/#gid0">
            <skos:prefLabel>Manchester</skos:prefLabel>
            <skos:definition>city</skos:definition>
          </rdf:Description>
        </rdf:li>
      </rdf:Bag>
    </skos:member>
    <skos:definition>country</skos:definition>
  </rdf:Description>
</rdf:RDF>
```

### Update or create a dataset

**PUT /metadata/dataset/** (*name*)

Updates or creates the existing dataset with the specified name.

**Query Parameters**

- **id-seed** (*string*) – A property name in the RDF model.

  If the id of a subject is not provided in the model, the value of this property will be used to generate an id for this subject.

  If not set or the subject doesn't have this property, a random id will be generated.

**Accepts**

- **application/rdf+xml, text/turtle, application/ld+json** – The RDF model.

**Produces**

- **application/rdf+xml, text/turtle, application/ld+json** – The RDF model.

**Role** _metadata_dataset_write

### Example

```
PUT /metadata/dataset/testmodel1?id-seed=skos:prefLabel
```

```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:skos="http://www.w3.org/2004/02/skos/core#">
   <rdf:Description>
     <skos:definition>country</skos:definition>
     <skos:member>
```

```
      <rdf:Bag>
        <rdf:li>
          <rdf:Description>
            <skos:definition>city</skos:definition>
            <skos:prefLabel>London</skos:prefLabel>
          </rdf:Description>
        </rdf:li>
        <rdf:li>
          <rdf:Description>
            <skos:definition>city</skos:definition>
            <skos:prefLabel>Manchester</skos:prefLabel>
          </rdf:Description>
        </rdf:li>
      </rdf:Bag>
    </skos:member>
    <skos:prefLabel>United Kingdom</skos:prefLabel>
  </rdf:Description>
</rdf:RDF>
```

Response:

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:skos="http://www.w3.org/2004/02/skos/core#"
    xmlns:vidi="http://rdf.vidispine.com/id/#">
  <rdf:Description rdf:about="http://rdf.vidispine.com/id/#unitedkingdom">
    <skos:prefLabel>United Kingdom</skos:prefLabel>
    <skos:member>
      <rdf:Bag rdf:about="http://rdf.vidispine.com/id/#gid0">
        <rdf:li>
          <rdf:Description rdf:about="http://rdf.vidispine.com/id/#london">
            <skos:prefLabel>London</skos:prefLabel>
            <skos:definition>city</skos:definition>
          </rdf:Description>
        </rdf:li>
        <rdf:li>
          <rdf:Description rdf:about="http://rdf.vidispine.com/id/#manchester">
            <skos:prefLabel>Manchester</skos:prefLabel>
            <skos:definition>city</skos:definition>
          </rdf:Description>
        </rdf:li>
      </rdf:Bag>
    </skos:member>
    <skos:definition>country</skos:definition>
  </rdf:Description>
</rdf:RDF>
```

### Delete a dataset

**DELETE /metadata/dataset/**(*name*)

Removes the metadata dataset with the specified name.

**Role** _metadata_dataset_write

## 17.16.13 Metadata migrations

Metadata migrations make it possible to update the metadata of existing items and collections to conform to a new metadata schema.

### Managing migrations

### List all metadata migrations

**GET /metadata/migration**
Lists all metadata migrations defined in the system.

**Role** _metadata_read

**Produces**

- **application/xml, application/json** – MetadataSchemaMigrationListDocument

### Create a metadata migration

**POST /metadata/migration**
Creates a new metadata migration.

**Role** _metadata_write

**Accepts**

- **application/xml, application/json** – MetadataSchemaMigrationDocument

**Produces**

- **text/plain** –

### Retrieve a metadata migration

**GET /metadata/migration/**(*id*)
Shows a single metadata migration.

**Role** _metadata_read

**Produces**

- **application/xml, application/json** – MetadataSchemaMigrationDocument

## 17.16.14 Metadata projections

A metadata projection is a bidirectional XSLT transformation, meant to simplify integration of the Vidispine system with several third party systems.

A projection consists of an incoming and an outgoing XSLT transformation.

- The incoming projection transforms information in some format to a format supported by Vidispine.
- The outgoing projection transforms information from Vidispine to a some other format.

When you use projections to transform item metadata then the outgoing projection will transform a MetadataListDocument and the incoming projection must produce a MetadataDocument.

### Get information about projections

### List all projections

**GET /projection**
Returns a list of all defined projections.

**Query Parameters**

- **url** (*boolean*) –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – CRLF-delimited list of ids or URLs

**Role** _projection_read

### Retrieve an outgoing projection

**GET /projection/** (*projection-id*) **/outgoing**
Returns the projection use to transform information *from* the Vidispine API, `GET metadata`.

**Status Codes**

- **404 Not found** – Could not find the projection identified by `projection-id`.

**Produces**

- **application/xml** – XML, XSLT stylesheet

**Role** _projection_read

### Retrieve an incoming projection

**GET /projection/** (*projection-id*) **/incoming**
Returns the projection use to transform information *to* the Vidispine API, `PUT metadata`.

**Status Codes**

- **404 Not found** – Could not find the projection identified by `projection-id`.

**Produces**

- **application/xml** – XML, XSLT stylesheet

**Role** _projection_read

### Create/modify/delete projections

**Note:** Please note that the projection result must be an valid XML document.

### Update or create an outgoing projection

**PUT /projection/** (*projection-id*) **/outgoing**
Creates a new projection if not defined earlier, and sets the outgoing projection to the specified stylesheet. If a new projection is created, the incoming transformation is set to be the identity transform.

**Accepts**

- **application/xml** – XML, XSLT stylesheet

**Produces**

- **application/xml** – XML, XSLT stylesheet

**Role** _projection_write

#### Update or create an incoming projection

**PUT /projection/** (*projection-id*) **/incoming**
Creates a new projection if not defined earlier, and sets the incoming projection to the specified stylesheet. If a new projection is created, the outgoing transformation is set to be the identity transform.

**Accepts**

- **application/xml** – XML, XSLT stylesheet

**Produces**

- **application/xml** – XML, XSLT stylesheet

**Role** _projection_write

#### Delete a projection

**DELETE /projection/** (*projection-id*)
Removes the projection.

**Status Codes**

- **200 OK** – The projection was deleted successfully.
- **404 Not found** – Could not find the projection identified by `projection-id`.

**Role** _projection_write

### 17.16.15 Metadata schema

A metadata schema can be used to enforce a particular data model in the metadata. A such restriction can say that the *field group* "goal" should contain exactly one *field* "goal_time" and one or more references to the group "player".

See *Defining a metadata schema* and *Alternate way of creating a schema* for an example on how to create a metadata schema.

#### Managing the metadata schema

#### Retrieve the schema

**GET /metadata-schema**
Retrieves the full metadata schema.

**Produces**

- **application/xml, application/json** – MetadataSchemaDocument

**Role** _metadata_schema_read

### Example

```
GET /metadata-schema
```

```
<MetadataSchemaDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <group min="0" max="-1" name="organization">
    <group reference="false" min="1" max="-1" name="employee"/>
    <group reference="false" min="0" max="-1" name="project"/>
    <field reference="false" min="1" max="1" name="example_name"/>
  </group>
  <group min="0" max="0" name="project">
    <group reference="true" min="1" max="-1" name="employee"/>
    <field reference="false" min="1" max="1" name="example_name"/>
    <field reference="false" min="1" max="1" name="example_location"/>
  </group>
  <group min="0" max="0" name="employee">
    <field reference="false" min="1" max="1" name="example_name"/>
    <field reference="false" min="0" max="1" name="example_title"/>
  </group>
</MetadataSchemaDocument>
```

### Update the schema

**PUT /metadata-schema**
Updates the schema with the given document.

**Accepts**

- **application/xml, application/json** – MetadataSchemaDocument

**Role** _metadata_schema_write

### Example

```
PUT /metadata-schema
Content-Type: application/xml

<MetadataSchemaDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <!-- The organization is optional and can exist [0,n] outside of groups -->
    <group name="organization" min="0" max="-1">
        <!-- An organization has one or more employees -->
        <group name="employee" min="1" max="-1" reference="false"/>
        <!-- An organization has one or more projects -->
        <group name="project" min="0" max="-1" reference="false"/>
        <!-- An organization has exactly one name -->
        <field name="example_name" min="1" max="1" reference="false"/>
    </group>

    <!-- A project cannot exist outside of a group -->
    <group name="project" min="0" max="0">
        <!-- A project has at least one employee, which has to be referenced -->
        <group name="employee" min="1" max="-1" reference="true"/>
        <!-- A project has exactly one name -->
        <field name="example_name" min="1" max="1" reference="false"/>
        <!-- A project has exactly one location element (it still can have more than␣
↪one value) -->
        <field name="example_location" min="1" max="1" reference="false"/>
```

```
    </group>

    <!-- An employee cannot exist outside of a group -->
    <group name="employee" min="0" max="0">
        <!-- An employee has exactly one name -->
        <field name="example_name" min="1" max="1" reference="false"/>
        <!-- An employee might have a title -->
        <field name="example_title" min="0" max="1" reference="false"/>
    </group>
</MetadataSchemaDocument>
```

### Delete the schema

**DELETE /metadata-schema**
Clears the schema, causing no validation to be made.

**Role** _metadata_schema_write

### Example

```
DELETE /metadata-schema
```

```
200 OK
```

### Groups in the schema

### Retrieve a group from the schema

**GET /metadata-schema/** (*group-name*)
Retrieves the schema for a particular group.

**Produces**

- **application/xml, application/json** – MetadataSchemaGroupDocument

**Role** _metadata_schema_read

### Example

```
GET /metadata-schema/project
```

```
<MetadataSchemaGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine" min="0
↪" max="0" name="project">
  <group reference="true" min="1" max="-1" name="employee"/>
  <field reference="false" min="1" max="1" name="example_name"/>
  <field reference="false" min="1" max="1" name="example_location"/>
</MetadataSchemaGroupDocument>
```

### Update a group in the schema

**PUT /metadata-schema/** (*group-name*)
Updates the specified group in the schema

**Accepts**

- **application/xml, application/json** – MetadataSchemaGroupDocument

**Role** _metadata_schema_write

#### Example

```
PUT /metadata-schema/employee
```

```
200 OK
```

#### Remove a group from the schema

**DELETE /metadata-schema/** (*group-name*)
Removes the group from the schema.

**Role** _metadata_schema_write

#### Example

```
DELETE /metadata-schema/employee
```

```
200 OK
```

### 17.16.16 Subtitles

Vidispine is able to produce SCC and TTML files from given subtitle metadata on items.

#### Exporting subtitles

#### Export to SCC

**GET /item/** (*id*) **/metadata/export/scc**
Returns the subtitles from the item metadata in SCC format.

**Query Parameters**

- **interval** (*string*) – Comma-separated list of *time spans* of subtitle metadata to export. Default is `all`, meaning all subtitles.

**Produces**

- **text/plain** – The SCC data.

**Role** _metadata_read

#### Example

```
PUT /item/VX-52/metadata/
```

```
<?xml version="1.0"?>
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="10" end="20">
    <group>
      <name>stl_subtitle</name>
      <field>
        <name>stl_text</name>
        <value>Text</value>
      </field>
```

```
    </group>
  </timespan>
</MetadataDocument>
```

```
GET /item/VX-56/metadata/export/scc
```

```
Scenarist_SCC V1.0

00:00:00:00  942c 942c 9420 9420 9470 9470 54e5 f8f4

00:00:10:00  942f 942f

00:00:20:00  942c 942c
```

### Export to TTML

**GET /item/** (*id*) **/metadata/export/ttml**
Generates a TTML file containing the subtitle metadata from a specific item.

The output TTML file obeys EBU-TT (http://tech.ebu.ch/ebu-tt) standard (EBU Tech 3360 (http://tech.ebu.ch/webdav/site/tech/shared/tech/tech3360.pdf))

An optional offset can be applied to the time span by adding ":" and the offset.

**Query Parameters**

- **interval** (*string*) – Comma-separated list of *time spans* of subtitle metadata to export. Default is `all`, meaning all subtitles.

**Produces**

- **application/xml** – The TTML XML.

**Role** _metadata_read

### Example

Firstly, import an STL file as a *sidecar* file

```
POST /import/sidecar/{item-id}?sidecar=subtitle.stl
```

Then export the STL metadata to TTML:

```
GET /item/{id}/metadata/export/ttml
```

Example result:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<tt xmlns:ns2="http://www.w3.org/ns/ttml#styling" xmlns="http://www.w3.org/ns/ttml"␣
↪xmlns:ns4="http://www.w3.org/ns/ttml#metadata" xmlns:ns3="urn:ebu:tt:style" xmlns:
↪ns5="http://www.w3.org/ns/ttml#parameter" xmlns:ns6="urn:ebu:tt:metadata" ns5:
↪frameRate="25" ns5:cellResolution="50 30" ns2:extent="704px 576px" xml:lang="en">
  <head>
    <metadata>
      <ns6:documentMetadata>
        <ns6:documentTargetAspectRatio>4:3</ns6:documentTargetAspectRatio>
        <ns6:documentTotalNumberOfSubtitles>11</ns6:documentTotalNumberOfSubtitles>
        <ns6:documentMaximumNumberOfDisplayableCharacterInAnyRow>40</ns6:
↪documentMaximumNumberOfDisplayableCharacterInAnyRow>
```

```
        <ns6:documentStartOfProgramme>00:00:00:00</ns6:documentStartOfProgramme>
        <ns6:documentCountryOfOrigin>GB</ns6:documentCountryOfOrigin>
        <ns6:documentPublisher>Institut fuer Rundfunktechnik   </ns6:
↪documentPublisher>
      </ns6:documentMetadata>
    </metadata>
    <styling>
      <style xml:id="textCenter" ns2:textAlign="center"/>
      <style xml:id="defaultStyle" ns2:fontFamily="monospaceSansSerif" ns2:fontSize=
↪"1c 1c" ns2:lineHeight="normal" ns2:textAlign="center" ns2:color="white" ns2:
↪backgroundColor="transparent" ns2:fontStyle="normal" ns2:fontWeight="normal" ns2:
↪textDecoration="none"/>
      <style xml:id="whiteOnblackDH" ns2:fontSize="1c 2c" ns2:color="white" ns2:
↪backgroundColor="black"/>
    </styling>
    <layout>
      <region xml:id="bottom" ns2:origin="10% 10%" ns2:extent="80% 80%" ns2:
↪displayAlign="after" ns2:padding="0c" ns2:writingMode="lrtb"/>
      <region xml:id="top" ns2:origin="10% 10%" ns2:extent="80% 80%" ns2:displayAlign=
↪"before" ns2:padding="0c" ns2:writingMode="lrtb"/>
    </layout>
  </head>
  <body>
    <div xml:id="SGN1" style="defaultStyle">
      <p region="top" style="textCenter" begin="00:00:00:00" end="00:00:02:10">
        <br/>
        <span style="whiteOnblackDH">two-line</span>
        <br/>
        <span style="whiteOnblackDH">top</span>
      </p>
      <p region="top" style="textCenter" begin="00:00:02:14" end="00:00:04:21">
        <br/>
        <span style="whiteOnblackDH">one-line top</span>
      </p>
      <p region="bottom" style="textCenter" begin="00:00:05:00" end="00:00:07:05">
        <span style="whiteOnblackDH">two-line</span>
        <br/>
        <span style="whiteOnblackDH">centre</span>
        <br/>
        <br/>
        <br/>
        <br/>
        <br/>
        <br/>
        <br/>
        <br/>
        <br/>
        <br/>
        <br/>
      </p>
      <p region="bottom" style="textCenter" begin="00:00:07:09" end="00:00:10:19">
        <span style="whiteOnblackDH">one-line centre</span>
        <br/>
        <br/>
        <br/>
        <br/>
        <br/>
        <br/>
        <br/>
```

```
        <br/>
        <br/>
        <br/>
      </p>
      <p region="bottom" style="textCenter" begin="00:00:14:06" end="00:00:17:10">
        <span style="whiteOnblackDH">two-line</span>
        <br/>
        <span style="whiteOnblackDH">bottom</span>
      </p>
      <p region="bottom" style="textCenter" begin="00:00:10:23" end="00:00:14:02">
        <span style="whiteOnblackDH">three-line</span>
        <br/>
        <span style="whiteOnblackDH">subtitle</span>
        <br/>
        <span style="whiteOnblackDH">bottom</span>
      </p>
      <p region="bottom" style="textCenter" begin="00:00:17:14" end="00:00:19:19">
        <span style="whiteOnblackDH">one-line bottom</span>
      </p>
      <p region="bottom" style="textCenter" begin="00:00:20:23" end="00:00:23:24">
        <span style="whiteOnblackDH">two-line subtitle</span>
        <br/>
        <span style="whiteOnblackDH">on row 16</span>
        <br/>
        <br/>
        <br/>
        <br/>
      </p>
      <p region="bottom" style="textCenter" begin="00:00:23:07" end="00:00:25:12">
        <span style="whiteOnblackDH">one-line row 18</span>
        <br/>
        <br/>
        <br/>
        <br/>
      </p>
      <p region="top" style="textCenter" begin="00:00:26:12" end="00:00:29:10">
        <br/>
        <br/>
        <br/>
        <br/>
        <br/>
        <span style="whiteOnblackDH">two-line subtitle</span>
        <br/>
        <span style="whiteOnblackDH">on row 5</span>
      </p>
      <p region="top" style="textCenter" begin="00:00:28:21" end="00:00:31:01">
        <br/>
        <br/>
        <br/>
        <br/>
        <br/>
        <span style="whiteOnblackDH"> one-line on row 5</span>
      </p>
    </div>
  </body>
</tt>
```

Only export the part of STL metadata to TTML:

```
GET /item/{id}/metadata/export/ttml?interval=905207@PAL-905253@PAL,904994@PAL-
↪905065@PAL
```

It is also possible to apply different offsets to the exported time intervals:

```
GET /item/{id}/metadata/export/ttml?interval=100@PAL-250@PAL:-25@PAL,500@PAL-1000@PAL:
↪-250@PAL
```

## 17.17 Miscellaneous

### 17.17.1 Stitching images

This resource provides an easy way of stitching images together. The result is cached in memory (and Memcached if it is activated in the configuration properties) for 3 hours.

It can be used with any image content and is not bound to be used with material imported in Vidispine. However, a typical usage of this service would be to assemble together representative thumbnails from several items of a collection.

**Note:** This request must go to http://server:8080/APInoauth/ instead of the usual http://server:8080/API/

**Note:** For a better quality rendering configure your Vidispine to use ImageMagick, see *Configuration properties*!

**Note:** Don't forget to URL encode all parameters!

#### Stitching images

#### Stitch images

**GET /APInoauth/stitch**

**Query Parameters**

- **uri** (*string[]*) – Multiple URI instances of the images to use. For example: `uri=img1.png&uri=img2.png&uri=img3.png...`
- **geometry** (*string*) – Geometry of the collage. For example, `100x50+10+10` will first downscale every image to fit in 100x50 then add a 10 pixels padding. If no parameter is specified, the original dimensions are used.
- **tile** (*string*) – The tiling strategy. Examples are `2x2` or `2x` or `x3`... If no parameter is specified, the system will try find the best alignment.
- **format** (*string*) – The format of the output image. Default is `PNG`.
- **background** (*string*) – The color of the background. Default is `white`. Color can be specified as hex values (`0xffffff`, `#ffffff`) or as name (`white`). `none` will create an image with transparent background (PNG only).

**Produces**

- **image/png , image/jpeg** – The stitched image

#### Example

```
GET /stitch?uri=http%3A%2F%2Fvidispine.com%2Fsites%2Fall%2Fthemes%2Fvidispine%2Flogo.
↪png&uri=http%3A%2F%2Fvidispine.com%2Fsites%2Fall%2Fthemes%2Fvidispine%2Flogo.png&
↪uri=http%3A%2F%2Fvidispine.com%2Fsites%2Fall%2Fthemes%2Fvidispine%2Flogo.png&
↪uri=http%3A%2F%2Fvidispine.com%2Fsites%2Fall%2Fthemes%2Fvidispine%2Flogo.png&
↪geometry=100x50%2B10&background=0xf5f7f8
```

### 17.17.2 Time zone

The `timezone` query parameter can be used to change the time zone of all dates in a response. This is supported for all API requests.

For example, to get the job information with dates in GMT-8.

```
GET /API/job?timezone=GMT-8
```

```
<JobListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <hits>116730</hits>
    <job>
      <jobId>VX-1</jobId>
      <user>admin</user>
      <started>2013-02-02T04:28:17.023-08:00</started>
      <status>ABORTED</status>
      <type>PLACEHOLDER_IMPORT</type>
      <priority>MEDIUM</priority>
    </job>
  ...
</JobListDocument>
```

**See also:**

TimeZone.getAvailableIDs (http://docs.oracle.com/javase/7/docs/api/java/util/TimeZone.html#getAvailableIDs()) for all legal time zone identifiers.

### 17.17.3 Troubleshooting

These resources are intended to aid when developing against Vidispine or when troubleshooting.

#### XML and JSON

#### Echo

**PUT /APInoauth/debug/echo**

Returns the provided XML in the Vidispine schema as either XML or JSON.

Use it to verify that XML is valid and parsed properly, or to convert from XML to JSON for example.

**Accepts**

- **application/xml** – Any XML in the Vidispine schema.

**Produces**

- **application/xml, application/json** – The input as XML or JSON.

#### Example

```
PUT /APInoauth/debug/echo
Accept: application/json
```

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <text>test</text>
</ItemSearchDocument>
```

```
200 OK
```

```
{
  "text": [
    {
      "value": "foo"
    }
  ]
}
```

### 17.17.4 WADL

A WADL file describing the API can be obtained using `GET API/application.wadl`.

**GET /application.wadl**
Retrieves the application WADL that describes the resources and methods exposed by the VS API.

**Produces**

- **application/vnd.sun.wadl+xml, application/xml** – The application WADL

### 17.17.5 Callback

#### CloudConvert Callback

**GET /APInoauth/callback/cloudconvert**
Used by CloudConvert to notify that a process has finished. See https://cloudconvert.com/api/conversions#callback.

Should not be used directly.

**Query Parameters**

- **id** (*string*) – The id of the CloudConvert process
- **url** (*string*) – location of the process result
- **step** (*string*) – The current step of the CloudConvert process

### 17.17.6 Return the current user

This resource will return the username of the calling user. This can for example be used to validate user credentials, or get the username of the caller when using token authorization.

**GET /whoami**
Validates the currently used credentials and returns the calling users username.

**Status Codes**

- **200 OK** – The credentials are correct.

- **401 Unauthorized** – The credentials are incorrect.

**Produces**

- **text/plain** – $USERNAME

## 17.18 Notifications

To define a notification, you need an action, a trigger, and a resource. Actions define what to do when the event happens. The trigger defines what type of event to trigger on, and the resource defines which entities to trigger for.

### 17.18.1 Actions

An action is what will be done when a notification is triggered. The action taken is that a message will be sent to either a HTTP, Java, Amazon SQS, Amazon SNS or JMS recipient, or being processed by a JavaScript.

An action can be sent either synchronously or asynchronously. The behaviour of synchronous vs. asynchronous notifications differ somewhat between the different action types, and is explained in the reference below. In general, however, synchronous notifications are sent from the same thread from where it was triggered, and asynchronous notifications are sent in a separate thread, allowing processing to continue right away after triggering.

#### Asynchronous delivery

Asynchronous notifications are delivered by multiple threads, one thread per notification endpoint (HTTP endpoint, Java method, JMS queue, SQS endpoint, SNS endpoint or script.) This can be customized by defining a thread group name for the actions. Notifications with the same thread group name will then be delivered by the same thread.

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        <http synchronous="false" group="app_notifications">
            ...
        </http>
    </action>
    <trigger>
        ...
    </trigger>
</NotificationDocument>
```

#### HTTP

Performs a HTTP request.

#### Action parameters

The details of the HTTP action are set in the action element in the notification definition.

**url** The URL to the HTTP resource. Mandatory parameter. Basic auth username and password is supported. (`http://user:password@host...`)

**method** The HTTP method to use, POST and PUT are supported. Mandatory parameter.

**timeout** The timeout (in seconds) before assuming network failure. Use 0 or none to specify that there is no timeout. Mandatory parameter.

**retry** The number of retries. Mandatory parameter.

**synchronous** See below. Mandatory parameter.

**contentType** The content type of the message sent to the destination. For supported values, see below. Optional value.

### Request body in notification message

The message in the request body depends on `contentType`:

**text/plain** Default format. A CRLF-delimited list with tab-separated rows that consist of the key followed by its values.

**application/xml** SimpleMetadataDocument

**application/json** SimpleMetadataDocument

### Error-handling logic

The output depends on the content-type set in the action definition. The way the HTTP action works depends on if it is setup as synchronous or asynchronous. Below is a table that shows the differences.

| Response | Synchronous | Asynchronous |
|---|---|---|
| Connection timeout | Stops | Will retry for a specified number of times |
| Receives HTTP code 2xx | Continues | Continues |
| Receives HTTP code 4xx | Stops | Stops |
| Receives HTTP code 5xx | Stops | Will retry for a specified number of times |

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        <http synchronous="false">
            <retry>3</retry>
            <contentType>application/json</contentType>
            <url>http://example.com/notify</url>
            <method>POST</method>
            <timeout>5</timeout>
        </http>
    </action>
    <trigger>
        ...
    </trigger>
</NotificationDocument>
```

### Java

Invokes a Java class method.

Methods in the Java classes must have the following signature and should not throw any exceptions. A null value as a response will always be regarded as that the message is *not* accepted and the action should stop. Note that returning the empty string is not the same as returning null, and will just be treated as an empty response.

```
public java.lang.String methodName(java.util.Map<java.lang.String, java.util.List
↪<java.lang.String>>)
```

**Note:** For classes to be discovered, Spring Context Indexer (https://mvnrepository.com/artifact/org.springframework/spring-context-indexer) must be used to index the components in any custom JAR files exposed to Vidispine.

**Note:** JNDI names must be prefixed by `vidibrain` (case insensitive). The interface name must match the class named plus the postfix `Remote`.

Changed in version 5.0: These are now plain Java class method invocations, as Vidispine no longer runs in an EJB container.

### Action parameters

The details of the Java class action are set in the action element in the notification definition.

**Note:** For legacy reasons, the name of the element is `ejb`.

**bean** The fully qualified name of the class. Mandatory parameter.

**method** The method name of the class. See above for method signature. Mandatory parameter.

**data** Key-value pairs with additional parameters that are passed to the bean method in the `java.util.Map<>` parameter. The keys in the map are the keys in the NotificationDocument prefixed by `data/` (e.g. the key `uri` in the NotificationDocument will get the key `data/uri` in the map).

**synchronous** See below. Mandatory parameter.

### Error-handling logic

| Response | Synchronous | Asynchronous |
|---|---|---|
| Could not find the bean | Stops | Continues |
| Could not find the method | Stops | Continues |
| Returns null value | Stops | Continues |
| Returns non-null value | Continues | Continues |

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <action>
      <ejb synchronous="true">
         <bean>vidibrain.beans.MyBeanRemote</bean>
         <method>myMethod</method>
         <data>
            <key>param1</key>
            <value>value1</value>
         </data>
         <data>
            <key>param2</key>
            <value>value2</value>
         </data>
         <data>
            <key>param3</key>
            <value>value3</value>
         </data>
      </ejb>
   </action>
   <trigger>
      ...
```

```
   </trigger>
</NotificationDocument>
```

### JMS

Sends a JMS message.

When running a standalone ActiveMQ instance, it is possible to send notifications to any queue that has been set up there. While it is possible to call them in asynchronous mode, there is not much point in doing so. This is since messages on JMS queues always are treated asynchronously. Below a table can be seen over what the outcome is based on the different responses.

#### Action parameters

The details of the JMS action are set in the action element in the notification definition.

**queue** The JNDI name of the JMS queue. Mandatory parameter.

**synchronous** See below. Mandatory parameter.

**contentType** The content type of the message sent to the destination. For supported values, see below. Optional value.

#### Notification message

The JMS message depends on `contentType`:

**application/x-java-serialized-object** Default format. An `ObjectMessage` with a `Map<String,List<String> >` object.

**text/plain** A `TextMessage` with a CRLF-delimited list with tab-separated rows that consist of the key followed by its values.

**application/xml** A `TextMessage` with SimpleMetadataDocument

**application/json** A `TextMessage` with SimpleMetadataDocument

#### Error-handling logic

| Response | Synchronous | Asynchronous |
|---|---|---|
| Could not find the queue | Stops | Continues |
| Could not find the queue factory | Stops | Continues |

#### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <action>
      <jms synchronous="true">
         <queue>MyNotificationQueue</queue>
      </jms>
   </action>
   <trigger>
      ...
   </trigger>
</NotificationDocument>
```

### JavaScript

Executes a JavaScript.

With a notification with a JavaScript action, it is possible to script actions directly in Vidispine. All of the output values (see below), are mapped to its respective variable in the JavaScript environment, unless if it is a multi-value list, then it is mapped to name + `List`. All output values are also available in the variable data, which is a Map from the id to a List of strings.

### Action parameters

The details of the JavaScript action are set in the action element in the notification definition.

**`script`** The actual JavaScript. Mandatory parameter.

**`synchronous`** See below. Mandatory parameter.

### Error-handling logic

| Response | Synchronous | Asynchronous |
|---|---|---|
| Errors (exceptions) in the execution | Stops | Continues |

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <action>
      <javascript>
         <script>
             logger.log("itemId: "+itemId+", "+data);
         </script>
      </javascript>
   </action>
   <trigger>
      ...
   </trigger>
</NotificationDocument>
```

### Amazon SQS

It is possible to send a message to Amazon SQS. Both standard and FIFO queues are supported.

New in version 5.2: Support for FIFO queues added.

### Prerequisites

An SQS queue (https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-configure-create-queue.html) must first be created to receive the messages. In order to send notifications to the queue, the user (as defined by the `accessKey` in the NotificationDocument) must have the permissions:

- `sqs:SendMessage`
- `sqs:GetQueueUrl`

New in version 21.3: Support for IAM roles added.

If a role is being used, the role must have the required permissions, and the user must have permission to assume the role (`sts:AssumeRole`).

### SQS parameters

The SQS notification can be configured using the following fields and attributes of the NotificationDocument:

**queue** The queue name. For FIFO queues the name has to end with `.fifo`.

Mandatory parameter.

**endpoint** The endpoint of the SQS.

Mandatory parameter.

**accessKey** AWS access key. Note that the `secret` field should not be empty if this field is set.

Optional parameter.

**secret** If `accessKey` is set, this field should be the AWS secret key. Otherwise, it should be the *alias of the secret* that contains the credentials.

Optional parameter.

**roleArn** The roleArn to define the IAM role to use, if using IAM role.

Optional parameter.

**roleSessionName** The role session name, if using IAM role.

Optional parameter. VidiCore will assign a name if not set.

**roleExternalId** The Amazon externalId (https://aws.amazon.com/blogs/security/how-to-use-external-id-when-granting-access-to-your-aws-resources/) to use if using IAM role.

Optional parameter. Required only if `roleArn` is set and the specified role is configured with an `sts:ExternalId` condition

**synchronous** Mandatory parameter.

*More info about synchronous/asynchronous notification.*

**contentType** The content type of the message sent to the destination. For supported values, see below.

Optional value.

**messageGroupId** The id that specifies that a message belongs to a specific message group. If you don't need multiple ordered message groups, specify the same message group ID for all your messages.

For FIFO queues this parameter is mandatory. For standard queues this parameter is forbidden

New in version 5.2.

### Request body in notification message

The message in the request body depends on `contentType`:

**text/plain** Default format. A CRLF-delimited list with tab-separated rows that consist of the key followed by its values.

**application/xml** SimpleMetadataDocument

**application/json** SimpleMetadataDocument

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <action>
    <sqs synchronous="false">
      <contentType>application/xml</contentType>
      <endpoint>sqs.eu-west-1.amazonaws.com</endpoint>
      <queue>notification</queue>
      <secret>aws</secret>
    </sqs>
  </action>
  <trigger>
    ...
  </trigger>
</NotificationDocument>
```

### Amazon SNS

New in version 5.2.

Sends a message to Amazon SNS.

### Prerequisites

An SNS topic (https://docs.aws.amazon.com/sns/latest/dg/sns-create-topic.html) must first be created to receive the notifications. In order to send SNS notifications, the user (as defined by the `accessKey` in the NotificationDocument) must have the permission:

- `sns:Publish`.

New in version 21.3: Support for IAM roles added.

If a role is being used, the role must have the required permissions, and the user must have permission to assume the role (`sts:AssumeRole`).

### SNS parameters

The SNS notification can be configured using the following fields and attributes of the NotificationDocument:

**`topic`** The topic ARN. Mandatory parameter.

**`endpoint`** The endpoint of the SNS.

Mandatory parameter.

**`accessKey`** AWS access key. Note that the `secret` field should not be empty if this field is set.

Optional parameter.

**`secret`** If `accessKey` is set, this field should be the AWS secret key. Otherwise, it should be the *alias of the secret* that contains the credentials.

Optional parameter.

**`roleArn`** The roleArn to define the IAM role to use, if using IAM role.

Optional parameter.

**`roleSessionName`** The role session name, if using IAM role.

Optional parameter. VidiCore will assign a name if not set.

**`roleExternalId`** The Amazon externalId (https://aws.amazon.com/blogs/security/how-to-use-external-id-when-granting-access-to-your-aws-resources/) to use if using IAM role.

Optional parameter. Required only if `roleArn` is set and the specified role is configured with an `sts:ExternalId` condition

**synchronous** Mandatory parameter.

*More info about synchronous/asynchronous notification.*

**contentType** The content type of the message sent to the destination. For supported values, see below.

Optional value.

#### Request body in notification message

The message in the request body depends on `contentType`:

**text/plain** Default format. A CRLF-delimited list with tab-separated rows that consist of the key followed by its values.

**application/xml** SimpleMetadataDocument

**application/json** SimpleMetadataDocument

#### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <action>
    <sns synchronous="false">
      <contentType>application/xml</contentType>
      <endpoint>sns.eu-west-1.amazonaws.com</endpoint>
      <topic>arn:aws:sns:eu-west-1:##############:sns-topic-name</topic>
      <secret>aws</secret>
    </sns>
  </action>
  <trigger>
    ...
  </trigger>
</NotificationDocument>
```

### 17.18.2 Triggers

A trigger is the event that will cause the notification to be sent. Different triggers exist for different resources. The trigger used determines what output that can be expected. Note that all keys will not necessarily be set and some keys may have more than one value.

#### Item triggers

#### Item create

| Resource | `/item` | |
|---|---|---|
| Parameters | - | |
| Output | `itemId` | The id of the created item |
| | `action` | The string “CREATE” |

Notifies when an item has been created.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <item>
            <create/>
        </item>
    </trigger>
</NotificationDocument>
```

### Item delete

| | | |
|---|---|---|
| Resource | /item | |
| Parameters | - | |
| Output | itemId | The id of the deleted item |
| | action | The string "DELETE" |

Notifies when an item has been deleted.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <item>
            <delete/>
        </item>
    </trigger>
</NotificationDocument>
```

### Item modify

| | | |
|---|---|---|
| Resource | /item | |
| Parameters | field | Specifies which fields to trigger on. |
| | interval | Specifies which intervals to trigger on. |
| | language | Specifies which languages to trigger on. |
| | track | Specifies which tracks to trigger on. |
| Output | itemId | The id of the modified item. |
| | changeSetId | The id of the metadata change set that was created. |
| | (field-name) | The value of the field with the name (field-name). |

Notifies when the metadata of an item has been modified. For the syntax of the parameters, please refer to *Retrieve metadata*.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
```

```
    <trigger>
        <metadata>
            <modify>
                <field>field_a, field_b</field>
                <language>en_*</language>
            </modify>
        </metadata>
    </trigger>
</NotificationDocument>
```

### Item create access

| Resource | /item | |
|---|---|---|
| Parameters | - | |
| Output | notificationType | The string “access”. |
| | notificationTrigger | The string “CREATE”. |
| | itemId | The id of the item that were assigned the access control. |
| | accessId | The id of the access control. |
| | permission | The level of permission granted by the access control. |
| | user | If the access control grants access to a particular user, this value is set. |
| | group | If the access control grants access to a particular group, this value is set. |

Sends a notification when an access control is created.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <access>
            <create/>
        </access>
    </trigger>
</NotificationDocument>
```

### Item delete access

| Resource | /item | |
|---|---|---|
| Parameters | - | |
| Output | notificationType | The string “access”. |
| | notificationTrigger | The string “DELETE”. |
| | itemId | The id of the item that were assigned the access control. |
| | accessId | The id of the access control. |
| | permission | The level of permission granted by the access control. |
| | user | If the access control grants access to a particular user, this value is set. |
| | group | If the access control grants access to a particular group, this value is set. |

Sends a notification an access control is deleted.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <access>
            <delete/>
        </access>
    </trigger>
</NotificationDocument>
```

### Item change access

| Resource | /item | |
|---|---|---|
| Parameters | - | |
| Output | notificationType | The string “access”. |
| | notificationTrigger | The string “CHANGE”. |
| | itemId | The id of the item that were assigned the access control. |
| | accessId | The id of the access control. |
| | permission | The level of permission granted by the access control. |
| | user | If the access control grants access to a particular user, this value is set. |
| | group | If the access control grants access to a particular group, this value is set. |

Sends a notification when an access control is changed.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <access>
            <change/>
        </access>
    </trigger>
</NotificationDocument>
```

### Item create shape

| Resource | /item | |
|---|---|---|
| Parameters | - | |
| Output | notificationType | The string “shape”. |
| | notificationTrigger | The string “CREATE”. |
| | itemId | The id of the item that the shape belongs to. |
| | shapeId | The id of the shape. |

Sends a notification when a shape is created.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <shape>
            <create/>
        </shape>
    </trigger>
</NotificationDocument>
```

### Item modify shape

| Resource | /item | |
|---|---|---|
| Parameters | - | |
| Output | notificationType | The string “shape”. |
| | notificationTrigger | The string “MODIFY”. |
| | itemId | The id of the item that the shape belongs to. |
| | shapeId | The id of the shape. |

Sends a notification when a shape is modified.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <shape>
            <modify/>
        </shape>
    </trigger>
</NotificationDocument>
```

### Item delete shape

| Resource | /item | |
|---|---|---|
| Parameters | - | |
| Output | notificationType | The string “shape”. |
| | notificationTrigger | The string “DELETE”. |
| | itemId | The id of the item that the shape belongs to. |
| | shapeId | The id of the shape. |

Sends a notification when a shape is deleted.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
```

```
        <shape>
            <delete/>
        </shape>
    </trigger>
</NotificationDocument>
```

### Collection triggers

### Collection create

| Resource | /collection | |
|---|---|---|
| Parameters | - | |
| Output | collectionId | The id of the created collection |
| | action | The string “CREATE” |

Notifies when a collection has been created.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <collection>
            <create/>
        </collection>
    </trigger>
</NotificationDocument>
```

### Collection delete

| Resource | /collection | |
|---|---|---|
| Parameters | - | |
| Output | collectionId | The id of the deleted collection |
| | action | The string “DELETE” |

Notifies when a collection has been deleted.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <collection>
            <delete/>
        </collection>
    </trigger>
</NotificationDocument>
```

### Collection metadata

| Resource | /collection | |
|---|---|---|
| Parameters | field | Specifies which fields to trigger on. |
| | interval | Specifies which intervals to trigger on. |
| | language | Specifies which languages to trigger on. |
| | track | Specifies which tracks to trigger on. |
| Output | collectionId | The id of the modified collection. |
| | changeSetId | The id of the metadata change set that was created. |
| | (field-name) | The value of the field with the name (field-name). |

Notifies when the metadata of a collection has been modified. For the syntax of the parameters, please refer to *Retrieve metadata*.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
      <collection>
        <metadata>
            <modify>
                <field>field_a, field_b</field>
                <language>en_*</language>
            </modify>
        </metadata>
      </collection>
    </trigger>
</NotificationDocument>
```

### Collection modify

| Resource | /collection | |
|---|---|---|
| Parameters | - | |
| Output | notificationType | The string “collection”. |
| | notificationTrigger | The string “MODIFY”. |
| | collectionId | The id of the collection that has changed. |

Sends a notification when the content of a collection has been modified.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <collection>
            <modify/>
        </collection>
    </trigger>
</NotificationDocument>
```

### Document triggers

### Document create

| Resource | /document | |
|---|---|---|
| Parameters | - | |
| Output | document | The name of the created document |
| | action | The string “CREATE” |

Notifies when a document has been created.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <document>
            <create/>
        </document>
    </trigger>
</NotificationDocument>
```

### Document delete

| Resource | /document | |
|---|---|---|
| Parameters | - | |
| Output | document | The name of the deleted document |
| | action | The string “DELETE” |

Notifies when a document has been deleted.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <document>
            <delete/>
        </document>
    </trigger>
</NotificationDocument>
```

### Document update

| Resource | /document | |
|---|---|---|
| Parameters | field | Specifies which fields to trigger on. |
| | interval | Specifies which intervals to trigger on. |
| | language | Specifies which languages to trigger on. |
| | track | Specifies which tracks to trigger on. |
| Output | document | The name of the modified document. |
| | changeSetId | The id of the metadata change set that was created. |
| | (field-name) | The value of the field with the name (field-name). |

Notifies when the document metadata has been modified. For the syntax of the parameters, please refer to *Retrieve metadata*.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
      <document>
        <modify>
          <field>field_a, field_b</field>
          <language>en_*</language>
        </modify>
      </document>
    </trigger>
</NotificationDocument>
```

### Group triggers

### Group create

| Resource | /group | |
|---|---|---|
| Parameters | - | |
| Output | group | The name of the created group. |
| | action | The string "CREATE" |
| | username | The name of the user that created the group. |

Sends a notification when a group is created.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <group>
            <create/>
        </group>
    </trigger>
</NotificationDocument>
```

### Group delete

| Resource | /group | |
|---|---|---|
| Parameters | - | |
| Output | group | The name of the deleted group. |
| | action | The string "DELETE" |
| | username | The name of the user that deleted the group. |

Notifies when a group has been deleted.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <group>
            <delete/>
        </group>
    </trigger>
</NotificationDocument>
```

### Group modify

| Resource | /group | |
|---|---|---|
| Parameters | - | |
| Output | group | The name of the modified group. |
| | action | The string "MODIFY" |
| | username | The name of the user that modified the group. |
| | mode | Either has the value "ADD" or "REMOVE" depending on the action taken on the group. |
| | affectedGroup | Contains the name of any affected group. |
| | affectedUser | Contains the name of any affected user. |

Notifies when the contents of a group has been modified.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <group>
            <modify/>
        </group>
    </trigger>
</NotificationDocument>
```

### Job triggers

### Normal job notifications

```
POST /job/notification
Content-Type: application/xml

<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <action>
       ...
   </action>
   <trigger>
      <job>
         <stop/>
```

```
      </job>
   </trigger>
</NotificationDocument>
```

This notification will be triggered when any job stops.

### Placeholder job notifications

One way job notifications differ from other notifications is that they can be created in advanced and then later be specified when starting a job. Note that a single job notification can be used for several jobs.

Creating a placeholder notification, that triggers when jobs stop:

```
POST /job/notification
Content-Type: application/xml

<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <action>
         ...
   </action>
   <trigger>
      <job>
         <stop/>
         <placeholder>true</placeholder>
      </job>
   </trigger>
</NotificationDocument>
```

```
VX-16
```

Using that notification when creating a new import job:

```
POST /import/?URL=http://example.com/video.avi&notification=VX-16
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan start="-INF" end="+INF">
      <field>
         <name>title</name>
         <value>My notification item!</value>
      </field>
   </timespan>
</MetadataDocument>
```

### Job create

| Resource | /job | |
|---|---|---|
| Parameters | - | |
| Output | jobId | The id of the created job. |
| | action | The string “CREATE”. |
| | status | The status of the job. |
| | type | The type of the job. |

Notifies when a job is created.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <job>
            <create/>
        </job>
    </trigger>
</NotificationDocument>
```

### Job stop

| Resource | /job | |
|---|---|---|
| Parameters | - | |
| Output | jobId | The id of the stopped job. |
| | action | The string “STOP”. |
| | status | The status of the job. |
| | type | The type of the job. |
| | currentStepNumber | The current step number. |
| | currentStepStatus | The status of the current step. |

Notifies when a job has stopped running, either successfully or unsuccessfully. Note that the output may also contain additional job specific data such as `itemId`, in the case of an import job.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <job>
            <stop/>
        </job>
    </trigger>
</NotificationDocument>
```

### Job update

| Resource | /job | |
|---|---|---|
| Parameters | - | |
| Output | jobId | The id of the stopped job. |
| | action | The string “UPDATE”. |
| | status | The status of the job. |
| | type | The type of the job. |
| | currentStepNumber | The current step number. |
| | currentStepStatus | The status of the current step. |

Notifies when the status of a job changes. Note that the output may also contain additional job specific data such as `itemId`, in the case of an import job.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <job>
            <update/>
        </job>
    </trigger>
</NotificationDocument>
```

### Job filtering

### Job types

Filter criteria can be added to job notifications in order to filter the jobs they trigger on.

| Name | Description |
|---|---|
| type | The type of the job |
| step | The step that the job is currently on |
| jobdata | A particular value in the job metadata. Job metadata consists of key value pairs and can be matched either by string comparison or regular expressions. |

### Example

```
POST /job/notification
Content-Type: application/xml

<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <action>
         ...
   </action>
   <trigger>
      <job>
         <stop/>
         <filter>
           <type>PLACEHOLDER_IMPORT</type>
         </filter>
      </job>
   </trigger>
</NotificationDocument>
```

this notification is triggered only when a PLACEHOLDER_IMPORT job stops.

**Note:** If you find notification filtering not working, please verify that there is no normal job notification existing.

### Job metadata

Either by string comparison or regular expressions:

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <action>
         ...
```

```
   </action>
   <trigger>
      <job>
         <stop/>
         <filter>
           <type>PLACEHOLDER_IMPORT</type>
           <jobdata>
                <key>key</key>
                <value>value</value>
           </jobdata>
         </filter>
      </job>
   </trigger>
</NotificationDocument>
```

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <action>
         ...
   </action>
   <trigger>
      <job>
         <stop/>
         <filter>
           <type>PLACEHOLDER_IMPORT</type>
           <jobdata>
                <key-regex>regex</key-regex>
                <value-regex>regex</value-regex>
           </jobdata>
         </filter>
      </job>
   </trigger>
</NotificationDocument>
```

### Content Filter

Content filter criteria can be added to job notifications in order to reduce the size of `jobDocument` returned by Vidispine.

Example:

```
POST /job/notification
Content-Type: application/xml

<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <action>
         ...
   </action>
  <trigger>
    <job>
      <stop/>
      <contentFilters>
        <contentFilter>jobId</contentFilter>
        <contentFilter>jobState</contentFilter>
      </contentFilters>
    </job>
  </trigger>
</NotificationDocument>
```

Legal content filters are:

```
<contentFilters>
  <contentFilter>jobId</contentFilter>
  <contentFilter>jobState</contentFilter>
  <contentFilter>user</contentFilter>
  <contentFilter>startTime</contentFilter>
  <contentFilter>jobType</contentFilter>
  <contentFilter>jobData</contentFilter>
  <contentFilter>errorMessage</contentFilter>
  <contentFilter>itemId</contentFilter>
  <contentFilter>totalSteps</contentFilter>
  <contentFilter>currentStep</contentFilter>
</contentFilters>
```

### Storage triggers

### Create storage

| Resource | /storage | |
|---|---|---|
| Parameters | - | |
| Output | storageId | The id of the created storage. |
| | action | The string “CREATE”. |

Notifies when a storage is created.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <storage>
            <create/>
        </storage>
    </trigger>
</NotificationDocument>
```

### Delete storage

| Resource | /storage | |
|---|---|---|
| Parameters | - | |
| Output | storageId | The id of the deleted storage. |
| | action | The string “DELETE”. |

Notifies when a storage is being deleted.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <storage>
```

```
            <delete/>
        </storage>
    </trigger>
</NotificationDocument>
```

### Generate filename

| Resource | /storage | |
|---|---|---|
| Parameters | - | |
| Output | storageId | The id of the storage. |
| | action | The string "FILENAME". |
| | itemId | The id of the item the file belongs to. |
| | shapeId | The id of the shape the file belongs to. |
| | componentId | The id of the component the file belongs to. |
| | fileId | The id of the file. |
| | username | The name of the user that causes the file to be created. |
| | metadata | The metadata of the item. |
| | shapeTag | A list of shape tags that belong to the shape. |

Notifies when a file is being created on a storage. The message returned by the HTTP or Java action will be used as a filename. If multiple notifications exist for a storage, then either one that returns a valid filename will be used. A valid filename follows the following format: `[A-Za-z0-9\_-]+`. It is recommended to use `fileId` in some way to guarantee the uniqueness of the filename.

Note that only the `storageId`, `action` and `fileId` output values are guaranteed to be included.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <storage>
            <filename/>
        </storage>
    </trigger>
</NotificationDocument>
```

**Note:** It is recommended to use the filename generation script function instead, see *Naming files on storage*.

### File triggers

Only generic file notifications are available, that is, there is no way to apply a notification to an individual file.

### Creating file notifications

Creating a placeholder notification, that triggers when file is created:

```
POST /storage/file/notification
Content-Type: application/xml

<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
```

```
   <action>
         ...
   </action>
   <trigger>
      <file>
         <new/>
      </file>
   </trigger>
</NotificationDocument>
```

```
VX-16
```

### New file

| Resource | /storage/file | |
|---|---|---|
| Parameters | - | |
| Output | fileId | The id of the new file. |
| | action | The string "NEW". |
| | storageId | The id of the storage the file is located on. |
| | shapeId | The shapeId of the file, if available. |
| | itemId | The id of the item the file is associated with if available. |

Notifies when a file is created, or has been discovered.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <file>
            <new/>
        </file>
    </trigger>
</NotificationDocument>
```

### File has changed

| Resource | /storage/file | |
|---|---|---|
| Parameters | - | |
| Output | fileId | The id of the changed file. |
| | action | The string "CHANGE". |
| | storageId | The id of the storage the file is located on. |
| | shapeId | The shapeId of the file, if available. |
| | itemId | The id of the item the file is associated with if available. |

Notifies when a file has changed.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
```

```
    </action>
    <trigger>
        <file>
            <change/>
        </file>
    </trigger>
</NotificationDocument>
```

### File has closed

| Resource | /storage/file | |
|---|---|---|
| Parameters | - | |
| Output | fileId | The id of the changed file. |
| | action | The string “CLOSE”. |
| | storageId | The id of the storage the file is located on. |
| | shapeId | The shapeId of the file, if available. |
| | itemId | The id of the item the file is associated with if available. |

Notifies when a file has been marked as closed in Vidispine.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <file>
            <close/>
        </file>
    </trigger>
</NotificationDocument>
```

### File was deleted

| Resource | /storage/file | |
|---|---|---|
| Parameters | - | |
| Output | fileId | The id of the deleted file. |
| | action | The string “DELETE”. |
| | storageId | The id of the storage the file is located on. |
| | shapeId | The shapeId of the file, if available. |
| | itemId | The id of the item the file is associated with if available. |

Notifies when a file has been deleted.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <file>
            <delete/>
```

```
        </file>
    </trigger>
</NotificationDocument>
```

### File hash has been computed

| Resource | /storage/file | |
|---|---|---|
| Parameters | - | |
| Output | fileId | The id of the hashed file. |
| | action | The string "HASH". |
| | storageId | The id of the storage the file is located on. |
| | shapeId | The shapeId of the file, if available. |
| | itemId | The id of the item the file is associated with if available. |

Notifies when a file has been hashed.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <file>
            <hash/>
        </file>
    </trigger>
</NotificationDocument>
```

### File has been lost

| Resource | /storage/file | |
|---|---|---|
| Parameters | - | |
| Output | fileId | The id of the hashed file. |
| | action | The string "LOST". |
| | storageId | The id of the storage the file is located on. |
| | shapeId | The shapeId of the file, if available. |
| | itemId | The id of the item the file is associated with if available. |

Notifies when a file has been lost.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <file>
            <lost/>
        </file>
    </trigger>
</NotificationDocument>
```

### Quota triggers

Quota triggers can be set to send a notification when a quota has been exceeded, when a quota is added, and when a quota is deleted.

### Create quota notifications

Creating a placeholder notification, that triggers when a quota is exceeded:

```
POST /quota/notification
Content-Type: application/xml

<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <action>
         ...
   </action>
   <trigger>
      <quota>
         <warning/>
      </quota>
   </trigger>
</NotificationDocument>
```

```
VX-16
```

### Quota warning

| Resource | /quota | |
|---|---|---|
| Parameters | - | |
| Output | ruleId | The id of the quota rule |
| | notificationTrigger | The string “WARNING”. |
| | itemLimit | The item limit set in the triggering rule. |
| | storageLimit | The storage limit set in the triggering rule. |
| | itemUsage | The current usage in items |
| | storageUsage | The current usage on storages. |

Notifies when a quota has been exceeded.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <quota>
            <warning/>
        </quota>
    </trigger>
</NotificationDocument>
```

### Quota create

| Resource | /quota | |
|---|---|---|
| Parameters | - | |
| Output | ruleId | The id of the quota rule |
| | notificationTrigger | The string “CREATE”. |
| | itemLimit | The item limit set in the triggering rule. |
| | storageLimit | The storage limit set in the triggering rule. |
| | itemUsage | The current usage in items |
| | storageUsage | The current usage on storages. |

Notifies when a quota has been created.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <quota>
            <created/>
        </quota>
    </trigger>
</NotificationDocument>
```

### Quota delete

| Resource | /quota | |
|---|---|---|
| Parameters | - | |
| Output | ruleId | The id of the quota rule |
| | notificationTrigger | The string “DELETE”. |
| | itemLimit | The item limit set in the triggering rule. |
| | storageLimit | The storage limit set in the triggering rule. |
| | itemUsage | The current usage in items |
| | storageUsage | The current usage on storages. |

Notifies when a quota has been exceeded.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <quota>
            <delete/>
        </quota>
    </trigger>
</NotificationDocument>
```

### Deletion lock triggers

New in version 4.15.

Changed in version 5.1.

The deletion-lock triggers now contains the `entityType` and `entityId` that the lock applies to.

### Lock create

| Resource | `.../deletion-lock` | |
|---|---|---|
| Parameters | - | |
| Output | `newLockId` | The id of the newly created lock. |
| | `newExpiry` | The expiration time of the lock. |
| | `entityType` | The type of the entity that the created lock applies to. |
| | `entityId` | The entity id that the created lock applies to. |

Triggered when a new deletion lock is created.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <action>
         ...
   </action>
   <trigger>
      <deletionLock>
         <create/>
      </deletionLock>
   </trigger>
</NotificationDocument>
```

### Lock modify

Triggered when a deletion lock has been modified.

| Resource | `.../deletion-lock` | |
|---|---|---|
| Parameters | - | |
| Output | `oldLockId` | The id of the lock that was modified. |
| | `oldExpiry` | The old expiration time of the lock. |
| | `newLockId` | The id of the lock that was modified. |
| | `newExpiry` | The new expiration time of the lock. |
| | `entityType` | The type of the entity that the modified lock applies to. |
| | `entityId` | The entity id that the modified lock applies to. |

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
      <deletionLock>
         <modify/>
      </deletionLock>
    </trigger>
</NotificationDocument>
```

### Lock delete

| Resource | .../deletion-lock | |
|---|---|---|
| Parameters | - | |
| Output | oldLockId | The id of the lock that was deleted. |
| | oldExpiry | The expiration time of that lock. |
| | entityType | The type of the entity that the deleted lock applied to. |
| | entityId | The entity id that the deleted lock applied to. |

Triggered when a deletion lock has been deleted, and the change has propagated to all the sub entities.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <deletionLock>
            <delete/>
        </deletionLock>
    </trigger>
</NotificationDocument>
```

### Lock expired

| Resource | .../deletion-lock | |
|---|---|---|
| Parameters | - | |
| Output | oldLockId | The id of the lock that expired. |
| | oldExpiry | The expiration time of that lock. |
| | entityType | The type of the entity that the expired lock applied to. |
| | entityId | The entity id that the expired lock applied to. |

Triggered when a lock has expired.

### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <deletionLock>
            <expire/>
        </deletionLock>
    </trigger>
</NotificationDocument>
```

#### Effective lock change

| Resource | .../deletion-lock | |
|---|---|---|
| Parameters | - | |
| Output | oldLockId | The id of the previous effective lock. |
| | oldExpiry | The old expiration time of the old lock. |
| | newLockId | The id of the new effective lock. |
| | newExpiry | The expiration time of the new lock. |
| | entityType | The type of the entity that the effective lock applies to. |
| | entityId | The entity id that the effective lock applies to. |

Triggered when an effective lock on the entity has changed.

#### Example

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <action>
        ...
    </action>
    <trigger>
        <deletionLock>
            <effective/>
        </deletionLock>
    </trigger>
</NotificationDocument>
```

### 17.18.3 Notifications

Most URLs that acts as resources in the API are used as resources in the notification framework as well.

In the following reference, {notification-entity} is one of the following:

- /item
- /collection
- /job
- /storage
- /storage/file
- /file
- /quota
- /group
- /document
- /deletion-lock

In the following reference, {notification-resource} is one of the following:

- {notification-entity}/notification
- /notification

Where the single notification endpoint is the placeholder-entity. To read and write these placeholder notifications the relevant role must be used, in this case the placeholder read/write roles.

In the following reference, {notification-entity-read-role} is one of the following:

- `_{notification-entity}_notification_read`
- `_placeholder_notification_read`

In the following reference, `{notification-entity-write-role}` is one of the following:

- `_{notification-entity}_notification_write`
- `_placeholder_notification_write`

Changed in version 4.17: The placeholder notification resource was added.

### Applying notifications to entire resources

Notifications can be applied to entire resources. For example if used on the item resource, then events that occur to any item will potentially trigger the event.

### List all notifications that exists within an entire resource

**GET {notification-resource}/**
Lists URIs to all notifications that exists within the given resource.

**Produces**

- **application/xml, application/json** – A URIListDocument containing URIs to all available notifications.
- **text/plain** – A CRLF-delimited list of URIs.

**Role** {notification-entity-read-role}

### Example

```
GET /item/job/notification
Accept: application/xml
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>VX-573</uri>
</URIListDocument>
```

### Create a notification that is applied to the entire resource

**POST {notification-resource}/**
Adds a notification that is applied to an entire resource

**Accepts**

- **application/xml, application/json** – NotificationDocument

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – The id of the notification.

**Role** {notification-entity-write-role}

### Example

Create a notification that triggers when PLACEHOLDER_IMPORT job finish.

```
POST /item/job/notification
Content-Type: application/xml

<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <action>
    <http>
      <url>http://10.10.0.3/notify-job</url>
      <timeout>5</timeout>
      <retry>3</retry>
      <method>POST</method>
      <contentType>application/xml</contentType>
    </http>
  </action>
  <trigger>
    <job>
      <finished/>
      <filter>
        <type>PLACEHOLDER_IMPORT</type>
      </filter>
    </job>
  </trigger>
</NotificationDocument>
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>VX-573</uri>
</URIListDocument>
```

### Delete all notifications that exists within an entire resource

**DELETE {notification-resource}/**

Removes all notifications that exists within the specified resource.

**Status Codes**

- **200 OK** – Everything went well.

**Role** {notification-entity-write-role}

### Retrieve a notification

**GET {notification-resource}/**(*notification-id*)

Retrieves a particular notification with the given id.

**Status Codes**

- **404 Not found** – No notification with that id exists in that resource.

**Produces**

- **application/xml, application/json** – NotificationDocument

**Role** {notification-entity-read-role}

### Example

Retrieve job notification VX-573.

```
GET /job/notification/VX-573
Accept: application/xml
```

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <action>
    <http>
      <url>http://10.10.0.3/notify-job</url>
      <timeout>5</timeout>
      <retry>3</retry>
      <method>POST</method>
      <contentType>application/xml</contentType>
    </http>
  </action>
  <trigger>
    <job>
      <finished/>
      <filter>
        <type>PLACEHOLDER_IMPORT</type>
      </filter>
    </job>
  </trigger>
</NotificationDocument>
```

### Delete a notification

**DELETE {notification-resource}/** (*notification-id*)
Removes a particular notification with the given id.

**Status Codes**

- **200 OK** – The notification was successfully removed.
- **404 Not found** – No notification with that id exists in that resource.

**Role** {notification-entity-write-role}

### Applying notifications to specific entities

Notifications can also be applied on specific entities, for example a single job.

**Note:** These resources do not apply for the `/storage/file` endpoint.

### List all notifications that exists for a particular entity

**GET {notification-entity}/** (*entity-id*) **/notification**
Lists URIs to all notifications that exists for a given entity.

**Produces**

- **application/xml, application/json** – A URIListDocument containing URIs to all available notifications.
- **text/plain** – A CRLF-delimited list of URIs.

**Role** {notification-entity-read-role}

### Create a notification that is applied to a particular entity

**POST {notification-entity}/**(*entity-id*)**/notification**
Adds a notification to the given entity.

**Accepts**

- **application/xml, application/json** – NotificationDocument

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – The id of the notification.

**Role** {notification-entity-write-role}

### Delete all notifications that is applied to a particular entity

**DELETE {notification-entity}/**(*entity-id*)**/notification**
Removes all notifications that exists within the specified entity.

**Status Codes**

- **200 OK** – Everything went well.

**Role** {notification-entity-write-role}

### Retrieve a notification

**GET {notification-entity}/**(*entity-id*)**/notification/**
*notification-id* Retrieves a particular notification with the given id.

**Status Codes**

- **404 Not found** – No notification with that id exists in that resource.

**Produces**

- **application/xml, application/json** – NotificationDocument

**Role** {notification-entity-read-role}

### Delete a notification

**DELETE {notification-entity}/**(*entity-id*)**/notification/**
*notification-id* Removes a particular notification with the given id.

**Status Codes**

- **200 OK** – The notification was successfully removed.
- **404 Not found** – No notification with that id exists in that resource.

**Role** {notification-entity-write-role}

### Change a particular notification on a whole entity

**PUT {notification-resource}/**(*notification-id*)
Change the action and/or trigger of the notification that exists within the given resource.

**Accepts**

- **application/xml, application/json** – NotificationDocument

**Produces**

- **application/xml, application/json** – NotificationDocument

**Role** {notification-entity-write-role}

**Status Codes**

- **200 OK** – The notification was successfully changed.
- **404 Not found** – No notification with that id exists in that resource.

### Change a particular notification on a specific entity

**PUT {notification-entity}/** (*entity-id*) **/notification/**
*notification-id* Change the action and/or trigger of the notification that exists within the given resource.

**Accepts**

- **application/xml, application/json** – NotificationDocument

**Produces**

- **application/xml, application/json** – NotificationDocument

**Role** {notification-entity-write-role}

**Status Codes**

- **200 OK** – The notification was successfully changed.
- **404 Not found** – No notification with that id exists in that resource.

### Example

```
GET /item/VX-43/notification/VX-573
```

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <action>
    <http>
      <url>http://10.10.0.3/notify-job</url>
      <timeout>5</timeout>
      <retry>3</retry>
      <method>POST</method>
      <contentType>application/xml</contentType>
    </http>
  </action>
  <trigger>
    <job>
      <finished/>
      <filter>
        <type>PLACEHOLDER_IMPORT</type>
      </filter>
    </job>
  </trigger>
</NotificationDocument>
```

```
PUT /item/VX-43/notification/VX-573
Content-Type: application/xml

<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
```

```
  <action>
    <http>
      <url>http://service.example.com/notify</url>
      <timeout>5</timeout>
      <retry>3</retry>
      <method>POST</method>
      <contentType>application/xml</contentType>
    </http>
  </action>
  <trigger>
    <job>
      <finished/>
      <filter>
        <type>TRANSCODE</type>
      </filter>
    </job>
  </trigger>
</NotificationDocument>
```

```
GET /item/VX-43/notification/VX-573

<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <action>
    <http>
      <url>http://service.example.com/notify</url>
      <timeout>5</timeout>
      <retry>3</retry>
      <method>POST</method>
      <contentType>application/xml</contentType>
    </http>
  </action>
  <trigger>
    <job>
      <finished/>
      <filter>
        <type>TRANSCODE</type>
      </filter>
    </job>
  </trigger>
</NotificationDocument>
```

## 17.19 Projects and versions

Manage projects.

### 17.19.1 Projects

#### Create a project collection

**POST /collection/project**

Creates a project collection with the given name and metadata.

**Accepts**

- **application/xml, application/json** – ProjectDocument

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – CRLF-delimited list of ids

#### Example

```
POST /collection/project
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProjectDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <name>Untitled Project</name>
  <metadata>...</metadata>
</ProjectDocument>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>VX-1</uri>
</URIListDocument>
```

### 17.19.2 Project versions

Manage project versions.

#### Create a project version collection

**POST /collection/**(*id*)**/version**

Creates a new project version collection for a specific project.

**Status Codes**

- **404 Not found** – Could not find the collection

**Accepts**

- **application/xml, application/json** – ProjectVersionDocument

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – CRLF-delimited list of ids

#### Example

```
POST /collection/VX-1/version
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProjectVersionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <item>
    <id>VX-1</id>
  </item>
  <sequence>...</sequence>
  <metadata>...</metadata>
</ProjectVersionDocument>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>VX-2</uri>
</URIListDocument>
```

### 17.19.3 Version definitions

#### List all project version definitions

**GET /collection/** (*id*) **/definition**

Returns the format of the definitions that have been stored for a specific project version.

**Status Codes**

- **404 Not found** – Could not find the collection

**Produces**

- **text/plain** – A CLRF separated list of formats.

**Role** _collection_read

#### Update a project version definition

**PUT /collection/** (*id*) **/definition/**

*format* Updates a binary representation of the project version. The collection must be a project version collection.

**Status Codes**

- **404 Not found** – Could not find the collection

**Accepts**

- **application/octet-stream** – The binary version definition

**Role** _collection_write

#### Retrieve a project version definition

**GET /collection/** (*id*) **/definition/**

*format* Retrieves a binary representation of the project version.

**Status Codes**

- **404 Not found** – Could not find the collection
- **404 Not found** – Could not find the definition

**Produces**

- **application/octet-stream** – The stored version definition

**Role** _collection_read

#### Delete a project version definition

**DELETE /collection/** (*id*) **/definition/**

*format* Deletes a binary representation of the project version.

**Status Codes**

- **404 Not found** – Could not find the collection

**Role** _collection_write

### 17.19.4 Assets in project version definition

To be able to track which clips and sequences corresponds to which items in Vidispine, an essence mapping can be stored for each version definition. This mapping is required for Vidispine to be able to convert between different formats.

#### Retrieve the asset definition

**GET /collection/** (*id*) **/definition/**

*format***/asset** Returns the asset document that has been stored for a specific project version definition.

**Status Codes**

- **404 Not found** – Could not find the collection
- **404 Not found** – Could not find the definition

**Produces**

- **application/xml, application/json** – EssenceMappingsDocument

**Role** _collection_read

#### Update the asset definition

**PUT /collection/** (*id*) **/definition/**

*format***/asset** Stores an asset document for a specific project version definition. The document should identify the items and files referenced by the definition.

**Status Codes**

- **404 Not found** – Could not find the collection
- **404 Not found** – Could not find the definition

**Accepts**

- **application/xml, application/json** – EssenceMappingsDocument

**Role** _collection_write

#### Delete the asset definition

**DELETE /collection/** (*id*) **/definition/**

*format***/asset** Deletes an asset document for a specific project version definition.

**Status Codes**

- **404 Not found** – Could not find the collection
- **404 Not found** – Could not find the definition

**Role** _collection_write

### 17.19.5 Version definition extradata

Each version definition can also store arbitrary extradata.

#### Retrieve the extradata

**GET /collection/** (*id*) **/definition/**
*format***/extradata** Returns the extradata that has been stored for a specific project version definition.

**Status Codes**

- **404 Not found** – Could not find the collection
- **404 Not found** – Could not find the definition

**Produces**

- **application/octet-stream** – The binary extradata.

**Role** _collection_read

#### Update the extradata

**PUT /collection/** (*id*) **/definition/**
*format***/extradata** Stores extradata for a specific project version definition.

**Status Codes**

- **404 Not found** – Could not find the collection
- **404 Not found** – Could not find the definition

**Accepts**

- **application/octet-stream** – The binary extradata.

**Role** _collection_write

#### Delete the extradata

**DELETE /collection/** (*id*) **/definition/**
*format***/extradata** Deletes the extradata for a specific project version definition.

**Status Codes**

- **404 Not found** – Could not find the collection
- **404 Not found** – Could not find the definition

**Role** _collection_write

### 17.19.6 Inspecting project files

#### Inspect a project file

**POST /collection/project/inspect**
Returns a document listing the sequences and assets referenced from a given project file.

**Query Parameters**

- **uri** (*string*) – The location of the file to inspect.
- **type** (*string*) – The file format.

**Accepts**

- **application/xml, application/json** – EssenceMappingsDocument

**Produces**

- **application/xml, application/json** – ProjectFileDocument

#### Example

```
POST /collection/project/inspect?uri=file:///home/maria/sequence.xml&amp;type=finalcut
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EssenceMappingDocument xmlns="http://xml.vidispine.com/schema/vidispine">
</EssenceMappingDocument>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProjectFileDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <location>file:///home/maria/sequence.xml</location>
  <asset>
    <id>urn:uuid:8CED8AFE-1A67-4632-AB57-D5F5B1E0BC49</id>
    <name>Sequence 1</name>
    <type>sequence</type>
    <status>unknown</status>
  </asset>
  <asset>
    <id>urn:uuid:FCAD0878-7129-43DA-A8A0-696590EFE4DA</id>
    <name>Sample Clip B</name>
    <type>clip</type>
    <status>unknown</status>
    <file>
      <path>file://localhost/Users/maria/Sample%20Clip%20B.mov</path>
    </file>
  </asset>
  <asset>
    <id>urn:uuid:76BE320F-48E0-47A5-A076-227158C50024</id>
    <name>Clip A</name>
    <type>clip</type>
    <status>unknown</status>
    <file>
      <path>file://localhost/Users/maria/Movies/Vidispine/VX-1.mov</path>
    </file>
  </asset>
</ProjectFileDocument>
```

### 17.19.7 Importing projects and sequences

#### Import a sequence

**POST /import/project/sequence**

Creates a new item with a Vidispine sequence representations of the given file. The file must contain a single sequence.

The item will also have a sequence contain the original data from the file, together with an essence mapping for identifying the items and files referenced by the sequence.

**Query Parameters**

- **uri** (*string*) – The location of the file to import.
- **type** (*string*) – The file format.
- **assignId** (*boolean*) – True if external ids should be created for the items in this project, using the ids from the project. Default is `false`.
- **pauseFrame** (*integer*) – When a rendering job is started, this parameter determines

which frame the job will pause at. The job will resume when the sequence is updated.

**Accepts**

- **application/xml, application/json** – EssenceMappingsDocument

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – CRLF-delimited list of ids

**Role** _import

**Role** _external_id_write

### Example

```
POST /import/project/sequence?uri=file:///home/maria/sequence.xml&amp;type=finalcut
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EssenceMappingsDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <asset id="urn:uuid:76BE320F-48E0-47A5-A076-227158C50024" item="VX-1"/>
  <file path="file://localhost/storage/VX-1.mov" hash=
↪"7b8d6ffe1ea468800578d6b7d4a09b012c461569"/>
</EssenceMappingsDocument>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>VX-8</uri>
</URIListDocument>
```

Retrieving the sequences:

```
GET /item/VX-8/sequence
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SequenceListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <sequence>
    <id>VX-1</sequence>
    <type>finalcut</type>
  </sequence>
  <sequence>
    <id>VX-2</sequence>
    <type>vidispine</type>
  </sequence>
</SequenceListDocument>
```

### Import a project

**POST /import/project**

Creates a new project version containing the clips and sequences found in the given project file. Sequences in the file will be created as items having a Vidispine sequence representation.

**Query Parameters**

- **collectionId** (*string*) – The id of the project to create a new version for.
- **uri** (*string*) – The location of the file to import.

- **type** (*string*) – The file format.
- **assignId** (*boolean*) – True if external ids should be created for the items in this project, using the ids from the project. Default is `false`.

**Accepts**

- **application/xml, application/json** – EssenceMappingsDocument

**Produces**

- **application/xml, application/json** – URIListDocument with the id of the new project version.
- **text/plain** – CRLF-delimited list of ids

**Role** _import

**Role** _collection_write

**Role** _external_id_write

### Example

Creating a new project:

```
POST /collection/project
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProjectDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <name>Marias Project</name>
  <metadata/>
</ProjectDocument>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>VX-1</uri>
</URIListDocument>
```

Creating a new project version with the clips and sequences from a Final Cut Pro project.

```
POST /import/project?collectionId=VX-1&amp;type=finalcut&amp;uri=file:///storage/
↪project.xml
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EssenceMappingsDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <asset id="urn:uuid:76BE320F-48E0-47A5-A076-227158C50024" item="VX-1"/>
  <file path="file://localhost/storage/VX-1.mov" hash=
↪"7b8d6ffe1ea468800578d6b7d4a09b012c461569"/>
</EssenceMappingsDocument>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>VX-2</uri>
</URIListDocument>
```

### 17.19.8 Exporting projects and sequences

#### Export a project or sequence

**POST /collection/** (*id*) **/version/export**
**POST /item/** (*id*) **/sequence/export**
**POST /sequence/export**

Exports the sequence or project to the requested output format and saves the result at the given location.

For POST /sequence/export the sequence must be specified in the request document.

**Query Parameters**

- **uri** (*string*) – The destination URI.
- **type** (*string*) – The output format.
- **tag** (*string*) – Comma separated list of shape tags specifying which shapes to reference in the output.
- **format** (*string*) – Comma separated list of formats specifying which shapes to reference in the output. If both tag and format is given, then both must match.
- **dryRun** (*boolean*) – When set to `true`, any export problems will be returned and no file will be written. Default is `false`
- **content** (*string*) – Comma-separated list of content to include in the output. Valid values are: `markers`, `subtitles`, `sequences`, `*` (default).
- **toSequence** (*boolean*) – Export as a sequence with an item, instead of as a standalone item. Default is `false`

**Status Codes**

- **404 Not found** – Invalid id

**Accepts**

- **application/xml, application/json** – ExportRequestDocument with details on the export.

**Produces**

- **application/xml, application/json** – ExportResponseDocument containing the essence mapping, and any export problems (if a dry run.)
- **multipart/mixed** – The response document and the export data. The uri parameter will be ignored.

The storage(s) that should be used can be specified in the request, and should be ordered in descending priority. For each storage it is also possible to specify the path to where the files will be available, so that formats that reference files by path can contain a usable client-side path.

#### Example

```
POST /item/VX-6/sequence/export?uri=file:///tmp/output.xml&amp;type=finalcut
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExportRequestDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <storage>
    <id>VX-1</id>
    <path>/Users/maria/Movies/Vidispine/</path>
```

```
  </storage>
</ExportRequestDocument>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExportResponseDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <mappings>
    <file path="file:///Users/maria/Movies/Vidispine/VX-1.mov" hash=
↪"7b8d6ffe1ea468800578d6b7d4a09b012c461569" size="30346173"/>
    <asset id="480a5bd6-b89a-476c-be2f-c2ce23ba53e8" item="VX-1"/>
  </mappings>
</ExportResponseDocument>
```

```
$ cat /tmp/output.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xmeml>
<xmeml version="5">
  ...
  <clip id="480a5bd6-b89a-476c-be2f-c2ce23ba53e8">
    <name>Item title</name>
    <uuid>480a5bd6-b89a-476c-be2f-c2ce23ba53e8</uuid>
    <file>
      ...
      <pathurl>file://localhost/Users/maria/Movies/Vidispine/VX-1.mov</pathurl>
    </file>
    ...
  </clip>
  ...
</xmeml>
```

Exporting to AAF without first transcoding the items to OP-Atom.

```
POST /item/VX-6/sequence/export?type=aaf&amp;dryRun=true&amp;tag=op-atom``
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExportRequestDocument xmlns="http://xml.vidispine.com/schema/vidispine">
</ExportRequestDocument>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExportResponseDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <problem>
    <type>missing-shape</type>
    <message>Found no shape matching the given format</message>
    <parameter>
      <key>item</key>
      <value>VX-1</value>
    </parameter>
    <parameter>
      <key>format</key>
      <value></value>
    </parameter>
    <parameter>
      <key>tag</key>
      <value>op-atom</value>
    </parameter>
```

```
  </problem>
  <mappings>
    <asset id="urn:uuid:11111111-2222-3333-4444-555555555555" item="VX-1"/>
  </mappings>
</ExportResponseDocument>
```

#### Export a project or sequence

**GET /collection/** (*id*) **/version/export**
**GET /item/** (*id*) **/sequence/export**

Exports the sequence or project to the requested output format and saves the result at the given location.

**Query Parameters**

- **type** (*string*) – The output format.
- **tag** (*string*) – Comma separated list of shape tags specifying which shapes to reference in the output.
- **format** (*string*) – Comma separated list of formats specifying which shapes to reference in the output. If both tag and format is given, then both must match.
- **dryRun** (*boolean*) – When set to `true`, any export problems will be returned and no file will be written. Default is `false`
- **content** (*string*) – Comma-separated list of content to include in the output. Valid values are: `markers`, `subtitles`, `sequences`, `*` (default).
- **storage** (*string*) – Comma-separated list of item paths in format `storageId=path`.
- **item** (*string*) – Comma-separated list of item paths in format `itemId=path`.
- **toSequence** (*boolean*) – Export as a sequence with an item, instead of as a standalone item. Default is `false`

**Status Codes**

- **404 Not found** – Invalid id

**Produces**

- **application/xml, application/json** – ExportResponseDocument if `dryRun=true`.
- ***/*** – The export data, in the media type of the format, if `dryRun=false`.

## 17.20 Quota rules

Vidispine can monitor the disk usage and the number of items that exist in the system, and send a notification if the usage exceeds certain user defined limits. A quota rule is used to define the limits that apply for a certain set of resources - items and files.

### 17.20.1 Managing quota rules

#### Create a quota rule

**POST /quota/**

Creates a quota rule with the filters and resource limits as specified in the quota rule document.

**Accepts**

- **application/xml, application/json** – QuotaRuleDocument

**Produces**

- **application/xml, application/json** – QuotaRuleDocument

**Role** _quota_write

### Example

Limit user `stephen` to 1000 items and 10000000 bytes of storage on VX-1:

```
POST /quota
Content-Type: application/xml

<QuotaRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <user>stephen</user>
  <storage>VX-1</storage>
  <resource>
    <name>item</name>
    <limit>1000</limit>
  </resource>
  <resource>
    <name>storage</name>
    <limit>10000000</limit>
  </resource>
</QuotaRuleDocument>
```

```
<QuotaRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-255</id>
  <user>stephen</user>
  <storage>VX-1</storage>
  <resource>
    <name>item</name>
    <limit>1000</limit>
    <usage>2</usage>
  </resource>
  <resource>
    <name>storage</name>
    <limit>10000000</limit>
    <usage>1266704</usage>
  </resource>
</QuotaRuleDocument>
```

### List all quota rules

**GET /quota/**

Returns the quota rules that exist in the system.

**Query Parameters**

- **content** (*string*) – Comma-separated list of addition content to retrieve. Valid values are: `external`.
- **filter** (*string*) – Comma-separated list of key-value pairs (in the format key=value) that can be used to filter the result set. Valid key values are: `user`, `group`, `storage`, `storageGroup`, `collection`, `library` and `tag`.
- **exceeded** (*boolean*) –

- `true` - Returns only rules where the usage of a resource exceeds the limit that has been specified for a rule.
- `false` (default) - Returns rules regardless of the current resource usage.

**Produces**

- **application/xml, application/json** – QuotaRuleListDocument

**Role** _quota_read

### Example

```
GET /quota
```

```
<QuotaRuleListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <rule>
    <id>VX-255</id>
    <user>stephen</user>
    <storage>VX-1</storage>
    <resource>
      <name>item</name>
      <limit>1000</limit>
      <usage>2</usage>
    </resource>
    <resource>
      <name>storage</name>
      <limit>10000000</limit>
      <usage>1266704</usage>
    </resource>
  </rule>
</QuotaRuleListDocument>
```

### Retrieve a quota rule

**GET /quota/** (*rule-id*)
Retrieves a specific quota rule.

**Produces**

- **application/xml, application/json** – QuotaRuleDocument

**Role** _quota_read

### Evaluate a quota rule

**PUT /quota/** (*rule-id*) **/evaluate**
Immediately calculates the resource usage as defined by a quota rule.

**Role** _quota_write

### Delete a quota rule

**DELETE /quota/** (*rule-id*)
Deletes the quota rule.

**Role** _quota_write

## 17.21 Resources

Manage external resources, such as transcoders or Vidinet services.

### 17.21.1 Resource types

#### List all resource types

**GET /resource**
Retrieves the available resource types.

**Produces**

- **application/xml, application/json** – ResourceTypeListDocument
- **text/plain** – CRLF-delimited list of resource type URLs ( `/resource/` { **type** })

**Role** _resource_read

### 17.21.2 Resources

#### List all resources

**GET /resource/** (*type*)
Retrieves the resources of the given type that have been configured.

**Produces**

- **application/xml, application/json** – ResourceListDocument
- **text/plain** – CRLF-delimited list of resource URLs ( `/resource/` { **type** } `/` { **resource-id** })

**Role** _resource_read

#### Create a resource

**POST /resource**
**POST /resource/** (*type*)
Create a new resource.

**Accepts**

- **application/xml, application/json** – ResourceDocument

**Produces**

- **application/xml, application/json** – ResourceDocument
- **text/plain** – Resource URL

**Role** _resource_write

#### Modify a resource

**PUT /resource/** (*type*) **/**
*resource-id* Updates an existing resource.

**Accepts**

- **application/xml, application/json** – ResourceDocument

**Produces**

- **application/xml, application/json** – ResourceDocument
- **text/plain** – Resource URL

**Role** _resource_write

**Status Codes**

- **400 Invalid Input** – The resource type is not correct.

#### Retrieve a resource

**GET /resource/**(*type*)**/**
*resource-id* Retrieves information on a specific resource.

**Produces**

- **application/xml, application/json** – ResourceDocument

**Role** _resource_read

**Status Codes**

- **400 Invalid Input** – The resource type is not correct.

#### Delete a resource

**DELETE /resource/**(*type*)**/**
*resource-id* Deletes the resource. All connection from other resources to the resource will become invalid.

**Role** _resource_write

**Status Codes**

- **400 Invalid Input** – The resource type is not correct.

### 17.21.3 Resource status

#### View resource status

**GET /resource/**(*type*)**/**
*resource-id***/status** Retrieves the status of a specific resource.

**Produces**

- **text/plain** – Status string

**Status Codes**

- **400** – If the resource does not have a status.

**Role** _resource_read

### 17.21.4 Resource configuration

#### Pre-check configuration

**GET /resource/**(*type*)**/**
*resource-id***/configuration/pre-check** New in version 21.3.

Perform a pre-check configuration of this resource. Only possible for VidiNet resources.

**Query Parameters**

- **displayData** (*boolean*) –

- `true` - Include detailed data of what will be changed during configuration.
- `false` (default) - Do not include detailed data.

**Produces**

- **application/xml, application/json** – ServiceConfigurationResultDocument

**Role** _administrator

### Apply configuration

**PUT /resource/** (*type*) **/**
*resource-id***/configuration** New in version 21.3.

Apply configuration for this resource. Only possible for VidiNet resources.

**Query Parameters**

- **displayData** (*boolean*) –
  - `true` - Include detailed data of what will be changed during configuration.
  - `false` (default) - Do not include detailed data.

**Produces**

- **application/xml, application/json** – ServiceConfigurationResultDocument

**Role** _administrator

## 17.22 Scheduling requests

Some resources support that requests are scheduled for later processing. This is done by specifying the field `schedule` in the request header. The value should be an ISO-8601 compatible timestamp stating the earliest time the request should be processed.

If the specified timestamp already has occurred, the call will proceed as usual. Otherwise HTTP status code 202 (Accepted) will be returned together with the CRLF-delimited triple (timestamp, request id, request URI).

For example, retrieving all metadata fields at a later time:

```
GET /metadata-fields HTTP/1.1
Schedule: 2010-07-02T11:55:00+02:00
```

```
HTTP/1.1 202 Accepted

2010-07-02T11:55:00+02:00       802972  http://localhost:8080/API/scheduled-request/
↪802972
```

### 17.22.1 States of scheduled requests

There are four states that a scheduled request can be in.

**WAITING** The request has been scheduled and is waiting to be processed.

**SUCCESS** The request has been processed successfully, by receiving status code 200 (OK).

**CONNECTION_FAILURE** The request has been processed, but failed for unknown reasons.

**BAD_REQUEST** The request has been processed, but received an unexpected status code.

## 17.22.2 Managing scheduled requests

### List all scheduled requests

**GET /scheduled-request**
Retrieves all known scheduled requests for the current user.

**Query Parameters**

- **state** (*string*) – Retrieve requests belonging to a certain state.

**Produces**

- **application/xml, application/json** – ScheduledRequestListDocument

### Example

```
GET /scheduled-request?state=SUCCESS
```

```
<ScheduledRequestListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <scheduledRequest>
    <id>802972</id>
    <user>admin</user>
    <state>SUCCESS</state>
    <date>2010-07-02T11:55:00.000+02:00</date>
    <created>2010-07-02T11:54:16.161+02:00</created>
    <executed>2010-07-02T11:55:36.762+02:00</executed>
    <request>
      <uri>http://localhost:8080/API/metadata-field</uri>
      <method>GET</method>
    </request>
    <response>
      <statusCode>200</statusCode>
      <hasBody>true</hasBody>
      <contentType>application/xml</contentType>
    </response>
  </scheduledRequest>
</ScheduledRequestListDocument>
```

### Retrieve a scheduled request

**GET /scheduled-request/**(*request-id*)
Retrieves the request that matches the specified id.

**Produces**

- **application/xml, application/json** – ScheduledRequestDocument

### Example

```
GET /scheduled-request/802972
```

```
<ScheduledRequestDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>802972</id>
  <user>admin</user>
  <state>SUCCESS</state>
  <date>2010-07-02T11:55:00.000+02:00</date>
  <created>2010-07-02T11:54:16.161+02:00</created>
```

```
  <executed>2010-07-02T11:55:36.762+02:00</executed>
  <request>
    <uri>http://localhost:8080/API/metadata-field</uri>
    <method>GET</method>
  </request>
  <response>
    <statusCode>200</statusCode>
    <hasBody>true</hasBody>
    <contentType>application/xml</contentType>
  </response>
</ScheduledRequestDocument>
```

### Retrieve the response body

**GET /scheduled-request/** (*request-id*) **/response**

Retrieves the response body of the scheduled request. This can only be called if the state of the request is either SUCCESS or BAD_REQUEST. The content-type is the same that was returned when the request was processed.

**Produces**

- ***/*** – The response body.

### Example

```
GET /scheduled-request/802972/response
```

```
<MetadataFieldListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field system="true">
    <name>durationTimeCode</name>
    <type>string-noindex</type>
  </field>
  <field system="true">
    <name>mimeType</name>
    <type>string-exact</type>
  </field>

...

</MetadataFieldListDocument>
```

### Delete all scheduled requests

**DELETE /scheduled-request/**

Deletes all scheduled requests for the current user.

### Example

```
DELETE /scheduled-request/
```

```
200 OK
```

### Delete a scheduled request

**DELETE /scheduled-request/** (*request-id*)

Deletes the scheduled request with the specified id.

#### Example

```
DELETE /scheduled-request/802972
```

```
200 OK
```

## 17.23 Search

Search for items and collections.

### 17.23.1 Search items and collections

Items and collections be browsed and searched with a single request. This type of search essentially has the combined set of the functionality of item search and collection search.

#### List all items and collections

**GET /search**

Browses items and collections.

**Content Parameters** See *Retrieving item information*

**Query Parameters**

- **first** (*integer*) – From the resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **cursor** (*string*) – New in version 5.4.
  - `*` - The initial cursor.
  - `string-from-search` - Cursor string returned from the search results.

  If set, the cursorMark (https://lucene.apache.org/solr/guide/6_6/pagination-of-results.html#fetching-a-large-number-of-sorted-results-cursors) / search after (https://www.elastic.co/guide/en/elasticsearch/reference/6.3/search-request-search-after.html#search-request-search-after) features from Solr/Elasticsearch are used to improve the deep paging (https://lucene.apache.org/solr/guide/6_6/pagination-of-results.html#performance-problems-with-deep-paging) performance during a search.

  When `cursor` is used, the value of `first` will be ignored.
- **number** (*integer*) – The number of entities to fetch. Default is `100`.
- **count** (*boolean*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **p** (*string*) – Comma-separated list of *paths* specifying the content to include. Overrides the content and filter parameters.
- **content** (*string*) – Comma-separated list of the types of content to retrieve. Valid values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.

  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **`field`** (*`string`*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **`group`** (*`string`*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **`language`** (*`string`*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **`sampleRate`** (*`string`*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **`track`** (*`string`*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 − t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **`include`** (*`string`*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **`includeValues`** (*`boolean`*) – Return the value enumeration for each metadata field.
- **`conflict`** (*`string`*) –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **`terse`** (*`string`*) –
  - `yes` - Return metadata in *terse format*.

  - `no` (default) - Return metadata in verbose format.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** (*boolean*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.
- **revision** (*string*) – Specifying which metadata revision to display. Only used if requesting a single item or collection.
- **mergedType** (*string*) – The type of operation to check for.
- **mergedPermission** (*string*) – The lowest required permission level.
- **mergedExtradata** (*string*) – Any possible extra data.
- **uriType** (*string*) – Comma-separated list of format types (container format) to return.
- **scheme** (*string*) – URI scheme to return.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodType** (*string*) – *Access method*.
  - `AUTO` - Gives an APInoauth URI to the media. Access to file is tunneled through Vidispine.
  - `AZURE_SAS` - If the storage schema is azure:// you can get direct access to the media. The resulting URI will not tunnel through Vidispine but rather point directly to the media location at the azure storage.
- **methodMetadata** (*string*) – *Metadata* used with storage method.
- **tag** (*string*) – A *URI parameter*: Comma-separated list of *shape tags* to return.
- **version** (*string*) – Specifying which essence version to return for shapes. If special value `all`, display all versions. If special value `latest` (default), display latest version of shapes.
- **closedFiles** (*boolean*) – A *URI parameter*:
  - `true` (default) - Return only URIs that point to closed files.
  - `false` - Return all URIs.
- **storage** (*string[]*) – List of storage ids. Return only files from specific storages. Can be specified multiple times.
- **storageGroup** (*string*) – Storage group id. Return only files from storages specified in the storage group.
- **noauth-url** (*boolean*) –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **baseURI** (*string*) – Which base URI to use for the thumbnail URLs.
- **save** (*boolean*) –

- `true` - Returns a `303 See Other`, with a `Location` header containing an URI to fetch the search result
- `false` (default) - Returns a regular search result

**Produces**

- **`application/xml, application/json`** – SearchResultDocument

**Role** _item_search

**Role** _collection_read

**Role** _metadata_read (`content=metadata`)

**Role** _item_uri (`content=uri`)

**Role** _thumbnail_read (`content=poster` and `content=thumbnail`)

**Role** _access_control_read (`content=access` and `content=merged-access`)

**Role** _item_id_read (`content=external`)

### Search items and collections

**`PUT /search`**

Searches items and collections with a shared search query.

---

**Note:** This resource doesn't support joint search, use *`PUT /item`*, *`PUT /search/shape`* or *`PUT /search/file`* for item, shape or file joint search respectively.

---

**Content Parameters** See *Retrieving item information*

**Query Parameters**

- **`first`** (*`integer`*) – From the resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **`cursor`** (*`string`*) – New in version 5.4.
  - `*` - The initial cursor.
  - `string-from-search` - Cursor string returned from the search results.

  If set, the cursorMark (https://lucene.apache.org/solr/guide/6_6/pagination-of-results.html#fetching-a-large-number-of-sorted-results-cursors) / search after (https://www.elastic.co/guide/en/elasticsearch/reference/6.3/search-request-search-after.html#search-request-search-after) features from Solr/Elasticsearch are used to improve the deep paging (https://lucene.apache.org/solr/guide/6_6/pagination-of-results.html#performance-problems-with-deep-paging) performance during a search.

  When `cursor` is used, the value of `first` will be ignored.
- **`number`** (*`integer`*) – The number of entities to fetch. Default is `100`.
- **`count`** (*`boolean`*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **`p`** (*`string`*) – Comma-separated list of *paths* specifying the content to include. Overrides the content and filter parameters.

- **content** (*string*) – Comma-separated list of the types of content to retrieve. Valid values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** (*string*) – Comma-separated list
  - *time-span* - Filter out metadata, return only metadata for specified *time span*.
  - `generic` - Return all non-timed metadata.
  - `all` (default) - Return all metadata, same as `interval=generic,-INF-+INF`
  - `result` - Can be used when retrieving metadata from a search result. Will return *time spans* that overlap with the search result. (New in 5.5.)
- **field** (*string*) – Comma-separated list.
  - *field-name* - Return specified field.
  - *field-name* ":" *new-name* - Return specified field, renamed to a new name in return value.
  - "-" *field-name* - Exclude specified field.
  - (default) - Return all fields.
- **group** (*string*) – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.
  - *group-name* **:** *new-name* - Return specified group, renamed to a new name in return value.
  - − *group-name* - Exclude specified group.
  - (default) - Return all groups.
- **language** (*string*) – Comma-separated list.
  - *language-tag* - Return metadata for specific language, e.g. `en_US`. Wildcards may be used, e.g. `*_CA` for both Canadian French and Canadian English.
  - `none` - Return all metadata without language specification.
  - `all` (default) - Return all metadata, with or without language specification.
- **sampleRate** (*string*) – Convert all outgoing *time instants* to specified rate. *NB! Time codes* which cannot be expressed in an integer number of samples will be returned as a decimal number, with risk of losing precision.
- **track** (*string*) – Comma-separated list.
  - *track-type track-number* - Return metadata for specified track. Example of track is `A2`.
  - *track-type t1 − t2* - Return metadata for specified track interval, e.g. `A2-4`.
  - *track-type* `*` - Return metadata for all tracks of specified type, e.g. `A*`.
  - `generic` - Return all non-tracked metadata.
  - `all` (default) - All metadata, with or without track specification, are returned.
- **include** (*string*) – A list of keys. Includes additional *field specific data*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **includeValues** (*boolean*) – Return the value enumeration for each metadata field.
- **conflict** (*string*) –

  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **terse** (*string*) –
  - `yes` - Return metadata in *terse format*.
  - `no` (default) - Return metadata in verbose format.
- **defaultValue** (*boolean*) –
  - `true` - For unset fields, return *default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** (*boolean*) –
  - `true` (default) - Include *transient metadata*.
  - `false` - Do not include transient metadata in response.
- **revision** (*string*) – Specifying which metadata revision to display. Only used if requesting a single item or collection.
- **mergedType** (*string*) – The type of operation to check for.
- **mergedPermission** (*string*) – The lowest required permission level.
- **mergedExtradata** (*string*) – Any possible extra data.
- **uriType** (*string*) – Comma-separated list of format types (container format) to return.
- **scheme** (*string*) – URI scheme to return.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodType** (*string*) – *Access method*.
  - `AUTO` - Gives an APInoauth URI to the media. Access to file is tunneled through Vidispine.
  - `AZURE_SAS` - If the storage schema is azure:// you can get direct access to the media. The resulting URI will not tunnel through Vidispine but rather point directly to the media location at the azure storage.
- **methodMetadata** (*string*) – *Metadata* used with storage method.
- **tag** (*string*) – A *URI parameter*: Comma-separated list of *shape tags* to return.
- **version** (*string*) – Specifying which essence version to return for shapes. If special value `all`, display all versions. If special value `latest` (default), display latest version of shapes.
- **closedFiles** (*boolean*) – A *URI parameter*:
  - `true` (default) - Return only URIs that point to closed files.
  - `false` - Return all URIs.
- **storage** (*string[]*) – List of storage ids. Return only files from specific storages. Can be specified multiple times.
- **storageGroup** (*string*) – Storage group id. Return only files from storages specified in the storage group.
- **noauth-url** (*boolean*) –
  - `true` Return URIs that do not need authentication.

- `false` (default) Return normal URIs

- **baseURI** (*string*) – Which base URI to use for the thumbnail URLs.
- **save** (*boolean*) –
  - `true` - Returns a `303 See Other`, with a `Location` header containing an URI to fetch the search result
  - `false` (default) - Returns a regular search result

**Accepts**

- **application/xml, application/json** – ItemSearchDocument

**Produces**

- **application/xml, application/json** – SearchResultDocument

**Role** _item_search

**Role** _collection_read

**Role** _metadata_read (`content=metadata`)

**Role** _item_uri (`content=uri`)

**Role** _thumbnail_read (`content=poster` and `content=thumbnail`)

**Role** _access_control_read (`content=access` and `content=merged-access`)

**Role** _item_id_read (`content=external`)

### Example

```
PUT /search?content=metadata&field=title
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <field>
        <name>title</name>
        <value>Something</value>
    </field>
</ItemSearchDocument>
```

```
<SearchResultDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <hits>3</hits>
  <entry start="-INF" end="+INF" type="Item" id="DE-42">
    <item id="DE-42" start="-INF" end="+INF">
      <metadata>
        <revision>DE-278,DE-276,DE-277</revision>
        <timespan start="-INF" end="+INF">
          <field uuid="e527b7f3-1bfa-4067-8dde-753368c09617" user="admin" timestamp=
↪"2012-03-23T10:10:42.845+01:00" change="DE-278">
            <name>title</name>
            <value uuid="38609429-67d1-4357-980c-64e7559768ff" user="admin" timestamp=
↪"2012-03-23T10:10:42.845+01:00" change="DE-278">Something</value>
          </field>
        </timespan>
      </metadata>
    </item>
    <timespan start="-INF" end="+INF"/>
  </entry>
```

```
  <entry start="-INF" end="+INF" type="Collection" id="DE-13">
    <collection>
      <id>DE-13</id>
      <metadata>
        <revision>DE-273,DE-279</revision>
        <timespan start="-INF" end="+INF">
          <field uuid="c203856a-e8e3-4d50-8287-642821941791" user="admin" timestamp=
↪"2012-03-23T10:10:57.103+01:00" change="DE-279">
            <name>title</name>
            <value uuid="f80fb9a1-1eca-479a-8b1a-11d30f93fa5d" user="admin" timestamp=
↪"2012-03-23T10:10:57.103+01:00" change="DE-279">Something</value>
          </field>
        </timespan>
      </metadata>
    </collection>
    <timespan start="-INF" end="+INF"/>
  </entry>
  <entry start="-INF" end="+INF" type="Item" id="DE-37">
    <item id="DE-37" start="-INF" end="+INF">
      <metadata>
        <revision>DE-255,DE-280,DE-256</revision>
        <timespan start="-INF" end="+INF">
          <field uuid="f1ac0198-6b8f-43a0-9caa-e8c36e80eee8" user="admin" timestamp=
↪"2012-03-23T10:11:16.849+01:00" change="DE-280">
            <name>title</name>
            <value uuid="dcd6a3bb-bf70-4cb7-8d77-e2adfe1e3b1c" user="admin" timestamp=
↪"2012-03-23T10:11:16.849+01:00" change="DE-280">Something</value>
          </field>
        </timespan>
      </metadata>
    </item>
    <timespan start="-INF" end="+INF"/>
  </entry>
</SearchResultDocument>
```

### 17.23.2 Search shapes

#### Search shapes

**GET /search/shape**

**PUT /search/shape**

Searches shapes. Using GET is identical as performing a search with an empty search document.

**Query Parameters**

- **first** (*integer*) – From the resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** (*integer*) – The number of entities to fetch. Default is `100`.
- **count** (*boolean*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **content** (*string*) – Comma-separated list of the types of content to retrieve. One of `component`, `metadata`, `essenceVersion`, `tag`, `mimeType` and `*`.

- **methodType** (*string*) – Return URIs from storage methods with a particular *type*. By default, return URLs with empty `methodType`.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodMetadata** (*string[]*) – *metadata* used with storage method.
- **scheme** (*string*) – URI scheme to return.
- **save** (*boolean*) –
  - `true` - Returns a `303 See Other`, with a `Location` header containing an URI to fetch the search result
  - `false` (default) - Returns a regular search result

**Accepts**

- **application/xml, application/json** – ShapeSearchDocument

**Produces**

- **application/xml, application/json** – *ShapeListDocument*

**Role** _item_shape_read

### 17.23.3 Search files

#### Search files

**GET /search/file**

**PUT /search/file**

Searches files. Using GET is identical as performing a search with an empty search document.

---

**Note:** This resource searches the same index and will produce the same results as `GET /storage/(storage-id)/file`. The only difference is the syntax, that is, the query parameters versus search documents. When using a search document it is also possible to perform joins.

---

**Query Parameters**

- **first** (*integer*) – From the resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** (*integer*) – The number of entities to fetch. Default is `100`.
- **count** (*boolean*) –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **methodType** (*string*) – Return URIs from storage methods with a particular *type*. By default, return URLs with empty `methodType`.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodMetadata** (*string[]*) – *metadata* used with storage method.
- **scheme** (*string*) – URI scheme to return.
- **save** (*boolean*) –

- `true` - Returns a `303 See Other`, with a `Location` header containing an URI to fetch the search result
- `false` (default) - Returns a regular search result

**Accepts**

- **application/xml, application/json** – FileSearchDocument

**Produces**

- **application/xml, application/json** – FileListDocument

**Role** _file_read

### 17.23.4 Autocompletion

There are two ways to get autocomplete suggestions. The first way is to make a separate autocomplete request to the API, this will generate a result set spanning over all items that the user has access to. The second way is to embed the request in a search document. When performed in this way, the autocomplete suggestions will only span the matching result set.

#### Autocomplete text

Text can be autocompleted against the search index.

**PUT /search/autocomplete**

Requests suggestion matching text in all or a specific metadata field.

**Status Codes**

- **400 Bad request** – A parameter was invalid.

**Accepts**

- **application/xml, application/json** – AutocompleteRequestDocument

**Produces**

- **application/xml, application/json** – AutocompleteResponseDocument

**Role** _search

#### Example

Assuming the user intends to type "original duration". The user first starts typing "original":

```
PUT /search/autocomplete
Content-Type: application/xml

<AutocompleteRequestDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <text>orig</text>
  <maximumSuggestions>3</maximumSuggestions>
</AutocompleteRequestDocument
```

```
<AutocompleteResponseDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <suggestion>original</suggestion>
  <suggestion>origin</suggestion>
  <suggestion>originated</suggestion>
</AutocompleteResponseDocument>
```

Then the user continues to start typing "duration":

```
<AutocompleteRequestDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <text>original dur</text>
  <maximumSuggestions>3</maximumSuggestions>
</AutocompleteRequestDocument>
```

```
<AutocompleteResponseDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <suggestion>original duration</suggestion>
</AutocompleteResponseDocument>
```

### Autocomplete on one metadata field

Auto-complete on one metadata field is supported. In order to make the auto-complete case insensitive, the metadata field should be set as `<index>extend</index>`.

### Example:

A metadata field `foo_bar` with config:

```
<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
         <type>string-exact</type>
         <index>extend</index>
</MetadataFieldDocument>
```

and this filed contains multiple values: “Animal”, “Sky”, “Animal and Sky”, “animal and sky”

An auto-complete request with user input “animal a”:

```
PUT /search/autocomplete
Content-Type: application/xml

<AutocompleteRequestDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>foo_bar</field>
  <text>animal a</text>
</AutocompleteRequestDocument>
```

will give result:

```
<AutocompleteResponseDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <suggestion>Animal and Sky</suggestion>
  <suggestion>animal and sky</suggestion>
</AutocompleteResponseDocument>
```

### Autocompletion as part of a search

To perform an autocomplete within a search, just append one or more `<autocomplete>` elements in the search document. The syntax is the same as for a separate autocomplete request.

### Example

```
PUT /item
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>my_category</name>
    <value>stock_photo</value>
```

```
  </field>
  <autocomplete>
    <field>my_tag</field>
    <text>hi</text>
  </autocomplete>
</ItemSearchDocument>
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <hits>5</hits>
  <item id="VX-6934" start="-INF" end="+INF">
    <timespan start="-INF" end="+INF"/>
  </item>
  <item id="VX-3464" start="-INF" end="+INF">
    <timespan start="-INF" end="+INF"/>
  </item>
  <item id="VX-2658" start="-INF" end="+INF">
    <timespan start="-INF" end="+INF"/>
  </item>
  <item id="VX-7234" start="-INF" end="+INF">
    <timespan start="-INF" end="+INF"/>
  </item>
  <item id="VX-3723" start="-INF" end="+INF">
    <timespan start="-INF" end="+INF"/>
  </item>
  <autocomplete>
    <field>my_tag</field>
    <suggestion>highres</suggestion>
    <suggestion>hills</suggestion>
    <suggestion>history</suggestion>
  </autocomplete>
</ItemListDocument>
```

### 17.23.5 Optimize index

If you wish to optimize the Solr index then we recommend that this is done via this API instead of sending an optimize request to Solr directly.

---

**Note:** Solr needs twice the disk space when optimizing the index. Make sure there's enough free space on the drive hosting the Solr index before optimizing.

---

---

**Note:** Solr will not accept updates while optimizing, meaning that new or updated item will not appear in the search results until the optimize operation has finished.

---

#### Optimize the search index

**POST /search/optimize**

Submits an optimize request to Solr.

This request can be made to block until the optimize request has completed using the `blocking` parameter.

**Query Parameters**

- **blocking** (*boolean*) –
  - `true` - The request will block until the Solr optimize request has finished.

- `false` (default) - Optimize will be scheduled and the request will return immediately.

- **timeout** (*integer*) – Block for maximum number of specified milliseconds. Default is `0`, which means block indefinitely.

**Status Codes**

- **200 OK** – If the operation finished successfully. Only if `blocking=true`.
- **202 Accepted** – If the operation was accepted but not yet finished.
- **500 Internal Server Error** – If an error occurs.

**Role** _administrator

## 17.24 Self tests

Run system self-tests to check the status of the system.

### 17.24.1 Running the test

#### Execute all self tests

**GET /selftest**
Executes all the self tests.

**Query Parameters**

- **exclude** (*string*) – Comma-separated list of *test names* to exclude.

**Produces**

- **application/xml, application/json** – *SelfTestDocument*

**GET /APInoauth/selftest**
Executes a database test. Since database problems may cause API unavailable, there is a database test available on APInoauth.

**Query Parameters**

- **exclude** (*string*) – Comma-separated list of *test names* to exclude.

**Produces**

- **application/xml, application/json** – *SelfTestDocument*

#### Execute a self test

**GET /selftest/**(*test-name*)
Executes the test with the specified name.

**Produces**

- **application/xml, application/json** – *SelfTestDocument*

## 17.25 Shape tags

Shape tags define the available presets to use when transcoding.

### 17.25.1 Managing shape tags

#### List all shape tags

**GET /shape-tag/**
Retrieves all shape tags known by the system.

**Query Parameters**

- **url** (*boolean*) –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – A list of the tags.

**Role** _shape_tag_read

#### Update or create a shape tag

**PUT /shape-tag/**(*tag-name*)
Creates a new shape tag with the given tag name. If the tag already exists, its transcode preset will be updated.

**Accepts**

- **application/xml, application/json** – TranscodePresetDocument

**Role** _shape_tag_write

#### Example

Creating a shape tag that specifies FLV as the container format, FLV as the video codec and AAC as the audio codec and uses the face detect plugin.

```
PUT /shape-tag/my_flv
Content-Type: application/xml

<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <format>flv</format>
   <video>
      <codec>flv</codec>
   </video>
   <audio>
      <codec>aac</codec>
   </audio>
   <faceDetect>true</faceDetect>
</TranscodePresetDocument>
```

#### Retrieve a shape tag

**GET /shape-tag/**(*tag-name*)
Retrieves the transcode preset of shape tag with the given tag name.

**Produces**

- **application/xml, application/json** – TranscodePresetDocument

**Role** _shape_tag_read

#### Delete a shape tag

**DELETE /shape-tag/**(*tag-name*)
Deletes a shape tag with the given tag name.

**Status Codes**

- **200 OK** – Tag deleted successfully.
- **404 Not found** – No tag with that name exists.

**Role** _shape_tag_write

**Caution:** Note that the tag will also be removed from any existing shapes with which it is associated.

### 17.25.2 Tags of a shape

See *Tags of a shape* for how to manage the tags associated with a specific shape.

### 17.25.3 Transcode preset scripts

#### Retrieve the script for a shape tag

**GET /shape-tag/**(*tag-name*)**/script**
Retrieves the script of the shape tag.

**Produces**

- **application/javascript** – A JavaScript

**Role** _shape_tag_read

#### Update or create the script for a shape tag

**PUT /shape-tag/**(*tag-name*)**/script**
Sets a script for the shape tag.

**Accepts**

- **application/javascript** – A JavaScript

**Role** _shape-tag_write

#### Remove the script for a shape tag

**DELETE /shape-tag/**(*tag-name*)**/script**
Unsets the script for the shape tag.

**Role** _shape_tag_write

#### Test a script

**GET /shape-tag/**(*tag-name*)**/item/**
*item-id***/shape/***shape-id* Tests the script of the shape tag with the specified shape as input and returns the resulting preset.

**Query Parameters**

- **job** (*string*) – The id of a job to retrieve job metadata from.

**Produces**

- **application/xml, application/json** – TranscodePresetDocument

**Role** _shape_tag_read

## 17.26 Sites

Manage remote sites.

### 17.26.1 Managing sites

#### List all sites

**GET /site**

Retrieves the ids of the sites that have been configured.

**Query Parameters**

- **type** (*string*) –
  - `all` (default) - Return the ids of all added sites.
  - `current` - Return the id of the current site.

**Produces**

- **application/xml, application/json** – URIListDocument

#### Update or create a site

**PUT /site/** (*site-id*)

Adds a new site using the given site definition.

The only supported value for `syncPolicy` is currently `ONDEMAND`.

**Accepts**

- **application/xml, application/json** – SiteDefinitionDocument

**Produces**

- **application/xml, application/json** – SiteDefinitionDocument

#### Example

```
PUT /site/VY
Content-Type: application/xml

<SiteDefinitionDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <url>http://10.1.2.3:8080/API/</url>
   <username>site-manager</username>
   <password>p4ssw0rd</password>
   <syncPolicy>ONDEMAND</syncPolicy>
</SiteDefinitionDocument>
```

#### Retrieve a site

**GET /site/** (*site-id*)

Retrieves the definition for a specific site.

**Produces**

- **application/xml, application/json** – SiteDefinitionDocument

**Role** _site_manager

## 17.27 Site rules

### 17.27.1 Managing site rules

Site rules control the content that is synced with a remote site.

In the following reference, `{site-rule-entity}` is one of the following:

- `/item`
- `/item/{item-id}`
- `/collection`
- `/collection/{collection-id}`
- `/library`
- `/library/{library-id}`
- `/user`
- `/user/{username}`
- `/group`
- `/group/{group-name}`

#### List all site rules for an entity

**GET {site-rule-entity}/site-rule**
Retrieves all site rules for the given entity/entities.

**Produces**

- **application/xml, application/json** – SiteRuleListDocument

**Role** _site_rule_read

#### Example

```
GET /item/VX-62/site-rule
```

```
<SiteRuleListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <siteRule>
    <site>VY</site>
    <metadata>true</metadata>
    <access>true</access>
    <shape>original</shape>
  </siteRule>
  <siteRule>
    <site>VZ</site>
    <metadata>true</metadata>
    <access>true</access>
    <shape>lowres</shape>
    <shape>original</shape>
  </siteRule>
</SiteRuleListDocument>
```

### Create a site rule

**POST {site-rule-entity}/site-rule**
Creates a new site rule for an entity.

**Status Codes**

- **200 OK** – Rule set successfully.
- **400 Bad request** – The request was malformed.
- **404 Not found** – Could not find the specified entity.

**Accepts**

- **application/xml, application/json** – SiteRuleDocument

**Produces**

- **application/xml, application/json** – SiteRuleDocument

**Role** _site_rule_write

### Example

```
POST /item/VX-67/site-rule/
Content-Type: application/xml

<SiteRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <site>VY</site>
  <metadata>true</metadata>
  <access>true</access>
  <shape>original</shape>
  <shape>lowres</shape>
</SiteRuleDocument>
```

Set the site rule for users.

```
POST /user/site-rule/
Content-Type: application/xml

<SiteRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <site>VY</site>
</SiteRuleDocument>
```

Set the site rule for groups. Setting a generic site rule for groups will also enable syncing of all users.

```
POST /group/site-rule/
Content-Type: application/xml

<SiteRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <site>VY</site>
</SiteRuleDocument>
```

### Retrieve a site rule

**GET {site-rule-entity}/site-rule/**(*id*)
Retrieves a specific site rule.

**Produces**

- **application/xml, application/json** – SiteRuleDocument

**Role** _site_rule_read

#### Update a site rule

**PUT {site-rule-entity}/site-rule/**(*id*)
Updates a site rule.

**Status Codes**

- **200 OK** – Rule set successfully.
- **400 Bad request** – The request was malformed.
- **404 Not found** – Could not find the specified rule.

**Accepts**

- **application/xml, application/json** – SiteRuleDocument

**Produces**

- **application/xml, application/json** – SiteRuleDocument

**Role** _site_rule_write

#### Delete a site rule

**DELETE {site-rule-entity}/site-rule/**(*id*)
Deletes a site rule.

**Status Codes**

- **200 OK** – Rule deleted successfully.
- **400 Bad request** – The request was malformed.
- **404 Not found** – Could not find the specified rule.

**Role** _site_rule_write

## 17.28 Storages

### 17.28.1 Auto-import rules

Auto-import rules can be used to enable automatic import of new files added to a storage.

#### Reading/modifying auto-import rules

#### Set an auto-import rule

**PUT /storage/**(*storage-id*)**/auto-import/**
Sets the auto-import rule for the specified storage.

**Accepts**

- **application/xml, application/json** – AutoImportRuleDocument

**Role** _storage_write

**Note:** In order for auto imports to work the `showImportables` property must be set to `true` on the storage.

### Example

```
PUT /storage/VX-5/auto-import
Content-Type: application/xml

<AutoImportRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <metadata>
    <timespan start="-INF" end="+INF">
      <field>
        <name>title</name>
        <value>This is an auto-imported item.</value>
      </field>
    </timespan>
  </metadata>
  <tag>myflvtag</tag>
</AutoImportRuleDocument>
```

### Retrieve all auto-import rules

**GET /storage/auto-import/**
Returns all known auto-import rules.

**Produces**

- **application/xml, application/json** – AutoImportRuleListDocument

**Role** _storage_read

### Retrieve an auto-import rule

**GET /storage/** (*storage-id*) **/auto-import/**
Returns the auto-import rule for a storage if there is one.

**Produces**

- **application/xml, application/json** – AutoImportRuleDocument

**Role** _storage_read

### Disable an auto-import rule

**PUT /storage/** (*storage-id*) **/auto-import/disable**
Stops auto-import jobs from being created for new files on this storage.

**Status Codes**

- **404 Not found** – This storage does not have an auto-import rule.

**Role** _storage_write

### Enable an auto-import rule

**PUT /storage/** (*storage-id*) **/auto-import/enable**
Resumes creation of auto-import jobs for files on this storage.

**Status Codes**

- **404 Not found** – This storage does not have an auto-import rule.

**Role** _storage_write

#### Delete an auto-import rule

**DELETE /storage/** (*storage-id*) **/auto-import/**
Removes any auto-import rule that might exist on the storage.

**Role** _storage_write

### 17.28.2 Files

Manage files on a storage.

The base path, referred to as `{storage-resource}` below, is one of the following depending on if a request should resolve all files or files on a specific storage only.

- `/storage/{storage-id}`
- `/storage`

Operations on files, uses `{file-resource}` can be one of:

- `/file/{file-id}`
- `{storage-resource}/file/{file-id}`
- `{storage-resource}/file/path/{path}`
- `{storage-resource}/file/uri/{uri}`

#### Information about files in storage

#### List all files in a storage

**GET {storage-resource}/file**
Retrieves the files for all or a specific storage.

There is a limit on how many files that can be returned for each call to this method. To get all files, iterate the calls.

**Query Parameters**

- **id** (*string[]*) – If multiple `id` query parameters are specified only those files are returned. If no ids are specified, all files are returned.
- **path** (*string*) –
  - *path* - Return files under this sub-path to storage.
  - `/` (default) - Return all files.
- **prefix** (*boolean*) –
  - `true` - Also include file prefixes that matches the criteria
  - `false` (default) - Do not include file prefixes
- **ignorecase** (*boolean*) –
  - `true` - Search file path case insensitively
  - `false` - Search file path case sensitively
- **recursive** (*boolean*) –
  - `true` (default) - Return all files in tree.
  - `false` - Return only files directly under specified path.

- **wildcard** (*boolean*) –
  - `true` - Allow use of wildcards in path.
  - `false` (default) - No wildcard handling of path.
- **first** (*integer*) – From total list of files, start return list from specified number. Default is `0`.
- **cursor** (*string*) – New in version 4.16.
  - `*` - The initial cursor.
  - `string-from-search` - Cursor string returned from the search results.

  If set, the cursorMark (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#fetching-a-large-number-of-sorted-results-cursors) / search after (https://www.elastic.co/guide/en/elasticsearch/reference/6.8/search-request-search-after.html#search-request-search-after) features from Solr/Elasticsearch would be used to improve the deep paging (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#performance-problems-with-deep-paging) performance during a search.

  When `cursor` is used, The value of `first` will be ignored.

  Changed in version 5.5.

  Starting in 5.5, cursor is returned for the end of the result instead of null to enable tailing (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#tailing-a-cursor) search.
- **number** (*integer*) – Return a maximum of specified number of files. Default is `10`.
- **prefixFirst** (*integer*) – From total list of prefixes, start return list from specified number. Default is `0`.

  Note: this parameter has no effect if Elasticsearch is the search backend.
- **prefixNumber** (*integer*) – Return a maximum of specified number of prefixes. Default is `10`.
- **sort** (*string*) – Comma-separated list. Use as: fileId desc,size desc,state desc.
  - `fileId` [ `asc` (default) | `desc` ] (default) - Order results by file id.
  - `size` [ `asc` (default) | `desc` ] - Order results by file size (bytes).
  - `state` [ `asc` (default) | `desc` ] - Order results by *file state*.
  - `timestamp` [ `asc` (default) | `desc` ] - Order results by file timestamp.
  - `filename` [ `asc` (default) | `desc` ] - Order results by filename.
  - `extension` [ `asc` (default) | `desc` ] - Order results by file extension.
- **storage** (*string[]*) – List of storage ids. Return only files from specific storages. Same as `storage-id` in URL, but can be specified multiple times
- **storageGroup** (*string*) – Storage group id. Return only files from storages specified in the storage group
- **filter** (*string*) –
  - `all` (default) - Return all files
  - `item` Only return files associated with an item.
  - `noitem` Only return files not associated with any item.

- **hash** (*string[]*) – List of hash values. Only return files with specific hash value.
- **algorithm** (*string*) – Hash algorithm. Search for hash values used by specified algorithm
- **count** (*boolean*) –
  - `true` (default) - Return total number of hits in result
  - `false` - Do not return total number of hits in result, in order to produce results faster
- **state** (*string[]*) – Filter results by *file state*. Can be used multiple times to select several states.
- **methodType** (*string*) – Return URIs from storage methods with a particular *type*. By default, return URLs with empty `methodType`.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodMetadata** (*string[]*) – *metadata* used with storage method.
- **scheme** (*string*) – URI scheme to return.
- **includeItem** (*boolean*) –
  - `true` - Return associated items, shapes, and components.
  - `false` (default) - Do not return any information about associated items, shapes, and components.
- **excludeQueued** (*boolean*) –
  - `true` - Exclude the files that are queued for import
  - `false` (default) - Do not exclude the files that are queued for import

**Produces**

- **application/xml, application/json** – FileListDocument
- **text/plain** – CRLF-delimited list of file ids

**Role** _file_read

### File search example

For examples on how `recursive` and `wildcard` works together, see the following table:

| Path | Recursive | Wildcard | File on storage | | | | Note |
|---|---|---|---|---|---|---|---|
| | | | foo | foo/bar | foo/bar/baz | foo/bar/?az | |
| | false | - | Y | N | N | N | (1) |
| | true | - | Y | Y | Y | Y | |
| / | false | - | Y | N | N | N | |
| / | true | - | Y | Y | Y | Y | |
| /foo | false | - | Y | N | N | N | |
| /foo | true | - | Y | Y | Y | Y | |
| foo | false | - | Y | N | N | N | (2) |
| foo | true | - | Y | Y | Y | Y | |
| /foo/ | false | - | N | Y | N | N | |
| /foo/ | true | - | N | Y | Y | Y | |
| /foo/bar | false | - | N | Y | N | N | |
| /foo/bar | true | - | N | Y | Y | Y | |
| /foo/bar/ | - | - | N | N | Y | Y | |
| \foo\bar\ | - | - | N | N | N | N | (3) |
| /foo/bax///../bar/./ | - | - | N | N | Y | Y | |
| /foo/b?r | false | true | N | Y | N | N | (4) |
| /foo/b??r | false | true | N | N | N | N | |
| /foo/*r | false | true | N | Y | N | N | (5) |
| /foo/**r | false | true | N | Y | N | N | |
| /foo/bar/?az | false | false | N | N | N | Y | (6) |
| /foo/bar/?az | false | true | N | N | Y | Y | |
| /foo/bar/\?az | false | false | N | N | N | N | (7) |
| /foo/bar/\?az | false | true | N | N | N | Y | (8) |

Notes:

1. Empty string = /.
2. The initial / is implicit.
3. / is the path delimiter in Vidispine, also on Windows.
4. ? = exactly one.
5. * = 0 or more
6. ? is not officially supported, see *URI's, URL's, and Special Characters*.
7. With wildcard=false, \ is literal. However, \ is not officially supported, see *URI's, URL's, and Special Characters*.
8. With wildcard=true, use \ to quote (" \ " **has** to be quoted).

Legend:

**-** Does not matter for this example

**Y** is included in search result

**N** is not include in search result

### Prefix search example

Assuming there is a file app/Phone/Android/Oreo/GoolgePlay.txt inside a storage, the result of some prefix search requests will be as follows:

```
GET /storage/{storage-id}/file?prefix=true
```

```
<FileDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <file>
  ...
  </file>
  <prefixes>
    <prefix>apps/Phone/Android/Oreo/</prefix>
  </prefixes>
</FileDocument>
```

```
GET /storage/{storage-id}/file?prefix=true&recursive=false
```

```
<FileDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <file>
  ...
  </file>
  <prefixes>
    <prefix>apps/</prefix>
  </prefixes>
</FileDocument>
```

```
GET /storage/{storage-id}/file?prefix=true&recursive=false&path=apps/
```

```
<FileDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <!-- The file list will be empty in this case -->
  <prefixes>
    <prefix>apps/Phone/</prefix>
  </prefixes>
</FileDocument>
```

In the case of `wildcard=true` and `recursive=false`, only the levels before the first wildcard character will be returned. So in the example below, `apps/` will be returned, instead of `apps/Phone/` or `apps/Phone/Android/`

```
GET /storage/{storage-id}/file?prefix=true&recursive=false&wildcard=true&path=app/
↪*Oreo*
```

```
<FileDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <file>
  ...
  </file>
  <prefixes>
    <prefix>apps/</prefix>
  </prefixes>
</FileDocument>
```

### Retrieve a file

**GET {file-resource}**

Retrieves the information, such as file size, status and checksum, of a specific file.

**Query Parameters**

- **methodType** (*string*) – Return URIs from storage methods with a particular *type*. By default, return URLs with empty `methodType`.

- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodMetadata** (*string[]*) – *metadata* used with storage method.
- **scheme** (*string*) – URI scheme to return.
- **includeItem** (*boolean*) –
  - `true` - Return associated items, shapes, and components.
  - `false` (default) - Do not return any information about associated items, shapes, and components.
- **includeShapes** (*boolean*) – New in version 5.3.
  - `true` - Include shapes deduced by *File Analyze Jobs*. A *ShapeListDocument* will be included in the metadata with key *faj_included_shapes*. The format of the document respects the Accept header.
  - `false` (default) - Do not include shapes.

**Produces**

- **application/xml, application/json** – FileDocument
- **text/plain** – *Tabbed tuples* of file id, file path, file size, file status, timestamp

**Role** _file_read

### Upload a file to a storage

**POST {storage-resource}/file/data**

Creates a new file on a specific storage, with the given file data.

**Query Parameters**

- **transferPriority** (*integer*) – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers. A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **transferId** (*string*) – An id to assign the transfer to be able to refer to it.
- **path** (*string*) – The path of the file on the storage.
- **uri** (*string*) – The absolute file URI (should be relative to the URI of a storage method).
- **state** (*string*) – The state of the file.

**Status Codes**

- **400** – If the amount of data received does not match the given Content-Length header.

**Accepts**

- **application/octet-stream** – The raw essence data.

**Produces**

- **application/xml, application/json** – FileDocument
- **text/plain** – file id

**Role** _file_write

### File data

#### Retrieve the file data

**GET {file-resource}/data**
Retrieves the raw file data.

**Produces**

- **application/octet-stream** – The raw file data.

**Role** _file_read

#### Update or create file data

**POST {file-resource}/data**
Uploads the file data for a specific file, overwriting any existing file data for the file.

**Query Parameters**

- **transferPriority** (*integer*) – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers. A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **transferId** (*string*) – An id to assign the transfer to be able to refer to it.

**Status Codes**

- **400** – If the amount of data received does not match the given Content-Length header.

**Accepts**

- **application/octet-stream** – The raw file data.

**Produces**

- **application/xml, application/json** – FileDocument

**Role** _file_write

#### Generate temporary credentials

New in version 4.15.

**POST {file-resource}/uri**
Generates temporary access credentials that give either READ or WRITE access directly to the file. By default, if the file is on S3 or Azure, this will try to create a read-only pre-signed URL for the file; if this fail or if the file is on another type of storage, it will try to create a proxy URL (with direct access to the file).

---

**Note:** Please note that these generated credentials will only allow access to the exact same key / filename, to use the examples below it would be `image.jpg`. You will not be able to read or write another key / filename using these credentials.

---

##### Amazon S3

When using the s3 scheme there are certain prerequisites that need to be met regarding policies and trust relationships with the account in use.

- If you specify which role to use by setting the *stsAssumeRole* property, you must make sure that this role have the permission policy to be able to assume the role (`sts:AssumeRole`) and that the policy allows to get the role (`iam:GetRole`).

- If you want to rely on Vidispine to find a role attached to an EC2 instance profile (by leaving the *stsAssumeRole* property unset), you must make sure that the policy allows for getting the EC2 instance profile (`iam:GetInstanceProfile`) and to get the attached role (`iam:GetRole`) and also the permission to assume this role (`sts:AssumeRole`).

The permission policy for the role will only require `s3:GetObject` and `s3:PutObject` permissions to use the basic features. If you intend to use this for multipart uploads you might also want to add the permissions for `s3:ListMultipartUploadParts` and `s3:AbortMultipartUpload`. Finally this role will also need a trust relationship with an account with access to the storage(s) as an intersection is made to decide permissions in the end.

Vidispine will then create a custom policy to limit the credentials to either GET or PUT as requested.

---

**Note:** When using s3 scheme, by default, the duration of the temporary security credentials for the role in AWS lasts for one hour. In order to use longer durations, the maximum session duration in AWS has to be increased. To learn how to view the maximum value for your role, see View the Maximum Session Duration Setting for a Role (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use.html#id_roles_use_view-role-max-session) in the IAM User Guide.

---

---

**Note:** If the s3 scheme is used, any transfer in progress when the duration ends will be interrupted, unlike pre-signed s3 URL's that would allow the file to be fully downloaded even if the duration would expire during transfer.

---

**Query Parameters**

- **scheme** (*string*) – Determines which type of URL / URI to be returned.
  - `s3` Utilize AWS's AssumeRole to generate a temporary URI giving access to only the specific file. Can be used either by setting the *stsAssumeRole* property to specify which role to assume when generating the credentials OR by leaving this unset which will make Vidispine try to use a role from an EC2 instance profile. These will also look at the configuration property for *stsRegion* and use that region when making the call to the STS API.
  - `https` (default) - Generates a temporary pre-signed HTTPS URL for either S3 or Azure, or a proxy URL (based on the configuration property *apiNoauthUri*) if on another type of storage.
  - `http` - Same as `https` but will also allow HTTP URL's to be returned.
- **write** (*boolean*) – Sets permission to either READ or WRITE.
  - `true` - Will give credentials with access to write. This would also enable the optional permissions for `s3:ListMultipartUploadParts` and `s3:AbortMultipartUpload` if used together with the `s3` scheme.
  - `false` (default) - Will give credentials with access to read.
- **duration** (*integer*) – Optional, sets the duration of the temporary credentials in minutes. Default is set to 15 minutes and the maximum is 720.

  Changed in version 4.17.7: The minimum duration for pre-signed URLs is 1 minute and when using the S3 scheme the minimum is 15 minutes.
- **sse-kms** (*boolean*) – New in version 5.6.

Set this to enable the pre-signed url to write a SSE-KMS encrypted file. Please note that this will only work together with the `https` scheme.

- `true` - Enable SSE-KMS for the written file. The AWS managed CMK for S3 will be used to encrypt the file unless a `kmsKeyId` is specified.
- `false` (default) - The file will be written without SSE-KMS.

- **kmsKeyId** (*string*) – New in version 5.6.

  The id of a customer managed CMK to use in SSE-KMS

**Produces**

- **application/xml, application/json** – URIListDocument

**Role** _file_write

### Examples

To generate a writeable S3 URL with temporary credentials:

**Note:** The user id and secret key will be replaced with the generated temporary credentials in the returned URI.

```
POST /storage/file/VX-56/uri?scheme=s3&write=true
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>s3://ASIA....:...@example/image.jpg?sessionToken=...</uri>
</URIListDocument>
```

To generate a pre-signed HTTPS URL (or proxy URL depending on the type of storage):

```
POST /storage/file/VX-98/uri?scheme=https&write=true
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>https://example.blob.core.windows.net/image.jpg?sig=...</uri>
</URIListDocument>
```

## Importable files

### List all files that can be imported

**GET {storage-resource}/importable**
Retrieves a list of files, together with any found metadata, for files that do not belong to any component.

Same query parameters as for *GET {storage-resource}/file*.

**Query Parameters**

- **auto** (*boolean*) –
  - `true` - Return files that will be automatically imported due to auto-import rules.
  - `false` (default) - Do not return files that will be automatically imported.
- **excludeQueued** (*boolean*) –
  - `true` (default) - Exclude the files that are queued for import
  - `false` - Do not exclude the files that are queued for import

- **id** (*string[]*) – If multiple `id` query parameters are specified only those files are returned. If no ids are specified, all files are returned.
- **path** (*string*) –
  - *path* - Return files under this sub-path to storage.
  - `/` (default) - Return all files.
- **prefix** (*boolean*) –
  - `true` - Also include file prefixes that matches the criteria
  - `false` (default) - Do not include file prefixes
- **ignorecase** (*boolean*) –
  - `true` - Search file path case insensitively
  - `false` - Search file path case sensitively
- **recursive** (*boolean*) –
  - `true` (default) - Return all files in tree.
  - `false` - Return only files directly under specified path.
- **wildcard** (*boolean*) –
  - `true` - Allow use of wildcards in path.
  - `false` (default) - No wildcard handling of path.
- **first** (*integer*) – From total list of files, start return list from specified number. Default is `0`.
- **cursor** (*string*) – New in version 4.16.
  - `*` - The initial cursor.
  - `string-from-search` - Cursor string returned from the search results.

  If set, the cursorMark (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#fetching-a-large-number-of-sorted-results-cursors) / search after (https://www.elastic.co/guide/en/elasticsearch/reference/6.8/search-request-search-after.html#search-request-search-after) features from Solr/Elasticsearch would be used to improve the deep paging (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#performance-problems-with-deep-paging) performance during a search.

  When `cursor` is used, The value of `first` will be ignored.

  Changed in version 5.5.

  Starting in 5.5, cursor is returned for the end of the result instead of null to enable tailing (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#tailing-a-cursor) search.
- **number** (*integer*) – Return a maximum of specified number of files. Default is `10`.
- **prefixFirst** (*integer*) – From total list of prefixes, start return list from specified number. Default is `0`.

  Note: this parameter has no effect if Elasticsearch is the search backend.
- **prefixNumber** (*integer*) – Return a maximum of specified number of prefixes. Default is `10`.
- **sort** (*string*) – Comma-separated list. Use as: fileId desc,size desc,state desc.

  - `fileId` [ `asc` (default) | `desc` ] (default) - Order results by file id.
  - `size` [ `asc` (default) | `desc` ] - Order results by file size (bytes).
  - `state` [ `asc` (default) | `desc` ] - Order results by *file state*.
  - `timestamp` [ `asc` (default) | `desc` ] - Order results by file timestamp.
  - `filename` [ `asc` (default) | `desc` ] - Order results by filename.
  - `extension` [ `asc` (default) | `desc` ] - Order results by file extension.
- **storage** (*string[]*) – List of storage ids. Return only files from specific storages. Same as `storage-id` in URL, but can be specified multiple times
- **storageGroup** (*string*) – Storage group id. Return only files from storages specified in the storage group
- **filter** (*string*) –
  - `all` (default) - Return all files
  - `item` Only return files associated with an item.
  - `noitem` Only return files not associated with any item.
- **hash** (*string[]*) – List of hash values. Only return files with specific hash value.
- **algorithm** (*string*) – Hash algorithm. Search for hash values used by specified algorithm
- **count** (*boolean*) –
  - `true` (default) - Return total number of hits in result
  - `false` - Do not return total number of hits in result, in order to produce results faster
- **state** (*string[]*) – Filter results by *file state*. Can be used multiple times to select several states.
- **methodType** (*string*) – Return URIs from storage methods with a particular *type*. By default, return URLs with empty `methodType`.
- **storageType** (*string*) – Only return URIs for files from storages of this *type*.
- **methodMetadata** (*string[]*) – *metadata* used with storage method.
- **scheme** (*string*) – URI scheme to return.

**Produces**

- **application/xml, application/json** – An ImportableFileListDocument describing the job.

**Role** _storage_read

### Importing a file from a storage

### Import a file

**POST {file-resource}/import**

Starts a an import job that will import the specified file. Only files that do not belong to any components can be imported.

**Query Parameters**

- **filename** (*string*) – The original filename of the file.

- **allowReimport** (*boolean*) –
  - `true` - Import the file to this shape even if the file is already importing or is already part of another item.
  - `false` (default) Reject the request if the file with the given id has already been imported or is currently importing.
- **id** (*string*) – Comma-delimited list of external ids to assign to the item.
- **tag** (*string[]*) – A list of *shape tags* to use for transcoding.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **createThumbnails** (*boolean*) –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **no-transcode** (*boolean*) –
  - `true` - Will disable transcoding even if the `tags` parameter is set. Rather, the specified tag will be used to determine cropping, scaling etc. of thumbnails.
  - `false` (default) - Normal transcode.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **overrideFastStart** (*boolean*) –
  - `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
  - `false` - Do not use the transcoder's estimate of the duration.
- **requireFastStart** (*boolean*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** (*string*) – Estimated duration of the clip in seconds.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **settings** (*string*) – Pre-configured *import settings*.
- **importTag** (*string[]*) – A list of shape tags that the created shape will be associated with. Default is `original`.

**Accepts**

- **application/xml, application/json** – MetadataDocument, initial metadata that is given to the imported item.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

### Import an IMF package

**POST {file-resource}/import/assetmap**

Starts a an import job that will import an ASSETMAP file (SMPTE ST 429-9). Files pointed to by the assetmap (DCP/InterOp or SMPTE) has to be stored in the same directory.

Changed in version 5.3: IMF packages will now be validated using Photon and results saved as metadata on the item. Can be disabled with **jobMetadata** parameter.

**Query Parameters**

- **allowReimport** (*boolean*) –
  - true - Import the file to this shape even if the file is already importing or is already part of another item.
  - false (default) Reject the request if the file with the given id has already been imported or is currently importing.
- **id** (*string*) – Comma-delimited list of external ids to assign to the item.
- **tag** (*string[]*) – A list of *shape tags* to use for transcoding.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **createThumbnails** (*boolean*) –
  - true (default) - Generate thumbnails as per defined by shape tag.
  - false - Disable thumbnail generation.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **no-transcode** (*boolean*) –
  - true - Will disable transcoding even if the tags parameter is set. Rather, the specified tag will be used to determine cropping, scaling etc. of thumbnails.
  - false (default) - Normal transcode.
- **overrideFastStart** (*boolean*) –
  - true (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
  - false - Do not use the transcoder's estimate of the duration.
- **requireFastStart** (*boolean*) –
  - true (default) - Try to put the index tables (header) in front of the file.
  - false - Put header at end of file.
- **fastStartLength** (*string*) – Estimated duration of the clip in seconds.
- **notification** (*string*) – The *placeholder job notification* to use for this job.

- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **settings** (*string*) – Pre-configured *import settings*.
- **importTag** (*string[]*) – A list of shape tags that the created shape will be associated with. Default is `original`.

**Accepts**

- **application/xml, application/json** – MetadataDocument, initial metadata that is given to the imported item.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

The IMF import job accepts certain special **jobMetadata** parameters:

**skipImpValidation** If set to true, do not perform Photon IMF package validation.

New in version 5.3.

### Import a file using a path

**POST {storage-resource}/file/import**

Starts a an import job that will import the specified file. Only files that do not belong to any components can be imported.

**Query Parameters**

- **path** (*string*) – Required. The path of the file on the storage.
- **uri** (*string*) – The absolute file URI (should be relative to the URI of a storage method).
- **state** (*string*) – The state of the file. Default is `CLOSED`.
- **createOnly** (*boolean*) –
  - `true` - Fail if a file with that path already exists.
  - `false` (default) - Update the existing file if one exists, else create a new file entity.
- **filename** (*string*) – The original filename of the file.
- **allowReimport** (*boolean*) –
  - `true` - Import the file to this shape even if the file is already importing or is already part of another item.
  - `false` (default) Reject the request if the file with the given id has already been imported or is currently importing.
- **id** (*string*) – Comma-delimited list of external ids to assign to the item.
- **tag** (*string[]*) – A list of *shape tags* to use for transcoding.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.

- **createThumbnails** (*boolean*) –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **no-transcode** (*boolean*) –
  - `true` - Will disable transcoding even if the `tags` parameter is set. Rather, the specified tag will be used to determine cropping, scaling etc. of thumbnails.
  - `false` (default) - Normal transcode.
- **resourceId** (*string*) – The transcoder resource to use to execute the transcode.
- **overrideFastStart** (*boolean*) –
  - `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
  - `false` - Do not use the transcoder's estimate of the duration.
- **requireFastStart** (*boolean*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** (*string*) – Estimated duration of the clip in seconds.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **settings** (*string*) – Pre-configured *import settings*.
- **importTag** (*string[]*) – A list of shape tags that the created shape will be associated with. Default is `original`.

**Accepts**

- **application/xml, application/json** – MetadataDocument, initial metadata that is given to the imported item.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

### Import an IMF package using a path

**POST {storage-resource}/file/import/assetmap**

Starts a an import job that will import an `ASSETMAP` file (SMPTE ST 429-9). Files pointed to by the assetmap (DCP/InterOp or SMPTE) has to be stored in the same directory.

Changed in version 5.3: IMF packages will now be validated using Photon and results saved as metadata on the item. Can be disabled with **jobMetadata** parameter.

**Query Parameters**

- **path** (*string*) – Required. The path of the file on the storage.
- **uri** (*string*) – The absolute file URI (should be relative to the URI of a storage method).
- **state** (*string*) – The state of the file. Default is `CLOSED`.
- **createOnly** (*boolean*) –
  - `true` - Fail if a file with that path already exists.
  - `false` (default) - Update the existing file if one exists, else create a new file entity.
- **allowReimport** (*boolean*) –
  - `true` - Import the file to this shape even if the file is already importing or is already part of another item.
  - `false` (default) Reject the request if the file with the given id has already been imported or is currently importing.
- **id** (*string*) – Comma-delimited list of external ids to assign to the item.
- **tag** (*string[]*) – A list of *shape tags* to use for transcoding.
- **original** (*string*) – If specified, should be one of the tags specified in the tag parameter. Specifies that the original shape tag will be reset to the shape created to this tag.
- **createThumbnails** (*boolean*) –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **thumbnailService** (*string*) – Identifies which thumbnail resource that should be used.
- **createPosters** (*string*) – A list of *time codes* to use for creating posters.
- **no-transcode** (*boolean*) –
  - `true` - Will disable transcoding even if the `tags` parameter is set. Rather, the specified tag will be used to determine cropping, scaling etc. of thumbnails.
  - `false` (default) - Normal transcode.
- **overrideFastStart** (*boolean*) –
  - `true` (default) - Use transcoder's estimate of the duration for allocating header space in MOV files and similar files.
  - `false` - Do not use the transcoder's estimate of the duration.
- **requireFastStart** (*boolean*) –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** (*string*) – Estimated duration of the clip in seconds.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.

- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.
- **settings** (*string*) – Pre-configured *import settings*.
- **importTag** (*string[]*) – A list of shape tags that the created shape will be associated with. Default is `original`.

**Accepts**

- **application/xml, application/json** – MetadataDocument, initial metadata that is given to the imported item.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the import job.

**Role** _import

The IMF import job accepts certain special **jobMetadata** parameters:

**skipImpValidation** If set to true, do not perform Photon IMF package validation.

New in version 5.3.

### Move/copy/delete files from a storage

### Move/copy a file to another storage

**POST {file-resource}/storage/** (*target-storage-id*)

Starts a move or copy job for the specified file.

**Query Parameters**

- **move** (*boolean*) –
  - `true` - Delete the original file when the copy has finished.
  - `false` - Just copy the file, and leave the original.
- **filename** (*string*) – The desired target filename.
- **useOriginalFilename** (*boolean*) – If set to `true`, the file will keep its original filename if available. Default is `false`.
- **timeRequirement** (*integer*) – Number of seconds the target file is required to exist before being moved due to storage rules etc.
- **limitRate** (*integer*) – Throttle the rate at which the transfer takes place (bytes/second).
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _file_write

### Delete a file

**DELETE {file-resource}**
Starts a delete job for the specified file.

**Query Parameters**

- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _file_write

## Managing files

### Register a file

**POST {storage-resource}/file**
Creates a file entity in the database. Does not create any physical files. Either a storage id and path or an absolute URI may be given.

**Query Parameters**

- **path** (*string*) – The path of the file on the storage.
- **uri** (*string*) – The absolute file URI (should be relative to the URI of a storage method).
- **state** (*string*) – The state of the file. Default is `OPEN`.
- **createOnly** (*boolean*) –
  - `true` (default) - Fail if a file with that path already exists.
  - `false` - Update the existing file if one exists, else create a new file entity.

**Status Codes**

- **409 Conflict** – If `createOnly=true` and a file with that path already exists.

**Produces**

- **application/xml, application/json** – FileDocument
- **text/plain** – The id of the new or updated file.

**POST {storage-resource}/file**
Creates a file entity in the database. Does not create any physical files. Either a storage id and path or an absolute URI may be given.

**Query Parameters**

- **createOnly** (*boolean*) –
  - `true` (default) - Fail if a file with that path already exists.
  - `false` - Update the existing file if one exists, else create a new file entity.

**Status Codes**

- **409 Conflict** – If `createOnly=true` and a file with that path already exists.

**Accepts**

- **application/xml, application/json** – FileDocument with the path or URI and file state.

**Produces**

- **application/xml, application/json** – FileDocument
- **text/plain** – The id of the new or updated file.

### Unregister a file

**DELETE {file-resource}/entity**

Deletes a file entity from the database. Does not touch the physical file.

**Role** _file_write

### Register a new file path

**POST {file-resource}/path**

Registers a new file, with the new path, and change all relevant components to point to the new file instead. The old file is marked for deletion. Hence, caller should first do the physical move, then issue this command.

The path of file entities in Vidispine is immutable. This command is used when a file is moved manually (without Vidispine), and caller wants to register the new path.

Use the `duplicate` parameter to add another file as a duplicate. The file at the new location will be added to all components that the file is already a part of. No file entities will be removed.

**Query Parameters**

- **storage** (*string*) – The new storage, if omitted, the same storage.
- **path** (*string*) – Required. The new path.
- **state** (*string*) – New state of the file. (OPEN, CLOSED, etc).
- **duplicate** (*boolean*) –
  - `true` - The file at the target path is a duplicate of this file. The old file entity will NOT be removed.
  - `false` (default) - This target path is the new location. This old file entity will be removed.

**Produces**

- **application/xml, application/json** – FileDocument
- **text/plain** – File id

**Role** _file_write

### Update the file state

**PUT {file-resource}/state/** (*state*)

Changes the state of the specified file to the given state.

Can for example be used after writing a file to a storage, to immediately mark it as CLOSED and no longer growing.

**Parameters**

- **state** – The new state of the file.

**Produces**

- **application/xml, application/json** – The resulting FileDocument
- **text/plain** – The id of the file.

Example:

```
PUT /storage/file/VX-12/state/CLOSED
```

```
<FileDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-432</id>
  <path>sample.mov</path>
  <uri>file:///srv/media1/sample.mov</uri>
  <state>CLOSED</state>
  <size>7889811</size>
  <hash>5eb9646332c96c738b4cacf7bc110d4cd8523ed5</hash>
  <timestamp>2016-03-24T17:02:53.796+01:00</timestamp>
  <refreshFlag>1</refreshFlag>
  <storage>VX-1</storage>
 </FileDocument>
```

### Remove a file from an item

**PUT {file-resource}/abandon**
Disassociates (disconnects) the physical file from the item. The shape which the file resided in will still exist, but there is no longer any connection between the file and the shape or item.

**Query Parameters**

- **item** (*string*) – The item from which the file in unassociated

**Role** _file_write

### Set file hash

New in version 5.0.

**PUT {file-resource}/hash/** (*hash*)
Set a new hash value of a file.

**Query Parameters**

- **algorithm** (*string*) – Hash algorithm of the new hash. If omitted, the default SHA-1 hash is set.

**Role** _file_write

### Re-index a file

**PUT {file-resource}/re-index**
Queues a single file for re-index.

**Produces**

- **text/plain** –

See *Re-indexing metadata* if you wish to reindex all files in the system.

### Shape deduction on files

### Perform shape deduction on a file

**POST {file-resource}/analyze**

New in version 5.3.

Start a job that will deduce the shape of a file without importing it. The resulting shape can be found as a ShapeDocument in the jobmetadata and can be retrieved from the file when the job has completed. Shape deduction will still run as normal during import of the file even if it has already been performed on the file through this job. Running this job will not affect any shapes associated with this file that have been created trough other means (e.g. shape created by import). Note that the result from this shape deduction job is not as extensive as running an analyze job on an imported shape. See *Shape analysis*.

**Query Parameters**

- **resourceId** (*string*) – The transcoder resource to use to execute the analysis.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

**Role** _file_write

### Perform shape deduction on an IMF package

**POST {file-resource}/analyze/imp**

New in version 5.3.

Start a job that will deduce the shape of an IMF package without importing it. The file resource must be the ASSETMAP.xml. Files pointed to by the ASSETMAP has to be stored in the same directory. Multiple CPLs will produce multiple shapes. The resulting shapes can be found as a *ShapeListDocument* in the jobmetadata and can be retrieved from the ASSETMAP file and CPL files when the job has completed. Shape deduction will still run as normal during import of the file even if it has already been performed on the file through this job. Running this job will not affect any shapes associated with this file that have been created trough other means (e.g. shape created by import). Note that the result from this shape deduction job is not as extensive as running an analyze job on an imported shape. See *Shape analysis*.

**Query Parameters**

- **resourceId** (*string*) – The transcoder resource to use to execute the analysis.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _job_write

**Role** _file_write

#### Get shapes for a file

**GET {file-resource}/shape**
New in version 5.3.

Retrieve shapes that have been created by a shape deduction analysis of the file.

**Produces**

- **application/xml, application/json** – *ShapeListDocument*

**Role** _file_read

### 17.28.3 Storages

Manage storages.

#### Managing storages

#### List all storages

**GET /storage**
Retrieves the storages that have been configured.

**Query Parameters**

- **size** (*string*) – Range of bytes, in format *s1−s2*. Returns storages with nominal size that is in that range. Either number can be omitted to not specify lower/upper limit. *Size units* can be used. Default is −, all storages.
- **freebytes** (*string*) – Range of bytes, in format *s1−s2*. Returns storages with free space that is in that range. Either number can be omitted to not specify lower/upper limit. *Size units* can be used. Default is −, all storages.
- **usedbytes** (*string*) – Range of bytes, in format *s1−s2*. Returns storages with used space that is in that range. Either number can be omitted to not specify lower/upper limit. *Size units* can be used. Default is −, all storages.
- **freeamount** (*string*) – Range of percent as integers, in format *s1−s2*. Returns storages with used space that is in that range. Either number can be omitted to not specify lower/upper limit. Default is −, all storages.
- **files** (*string*) – Range of files as integers, in format *s1−s2*. Returns storages with number of files that is in that range. Either number can be omitted to not specify lower/upper limit. Default is −, all storages.
- **storagegroup** (*string[]*) – List of storage groups.
  - *storage-group* - Returned storage is member of specified storage group.
  - *−storage-group* - Returned storage is not member of specified storage group.

  Default is to return all storages.
- **status** (*string[]*) – List of *storage status*.
  - *status* - Returned storage has this status.
  - *−status* - Returned storage does not have this status.

Default is to return all storages.

- **url** (*string*) – Returns storages with a method matching this URL. May include wildcards * and ?.

**Produces**

- **application/xml, application/json** – StorageListDocument
- **text/plain** – CRLF-delimited list of storage ids

**Role** _storage_read

### Example

```
GET /storage
```

```
<StorageListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <storage>
    <id>VX-1</id>
    <state>NONE</state>
    <type>LOCAL</type>
    <capacity>1600000000000</capacity>
    <freeCapacity>19993896424</freeCapacity>
    <method>
      <id>VX-6</id>
      <uri>file:///mnt/main/</uri>
      <bandwidth>0</bandwidth>
      <read>true</read>
      <write>true</write>
      <browse>true</browse>
      <lastSuccess>2014-07-04T08:39:00.739+02:00</lastSuccess>
      <type>NONE</type>
    </method>
    <metadata/>
    <lowWatermark>0</lowWatermark>
    <highWatermark>0</highWatermark>
    <showImportables>true</showImportables>
  </storage>
  <storage>
    <id>VX-2</id>
    <state>NONE</state>
    <type>LOCAL</type>
    <capacity>15000000000</capacity>
    <freeCapacity>19548305962</freeCapacity>
    <method>
      <id>VX-4</id>
      <uri>file:///mnt/ingest/</uri>
      <read>true</read>
      <write>true</write>
      <browse>true</browse>
      <lastSuccess>2014-07-04T08:38:56.773+02:00</lastSuccess>
      <type>NONE</type>
    </method>
    <metadata/>
    <lowWatermark>0</lowWatermark>
    <highWatermark>200000000000</highWatermark>
    <showImportables>true</showImportables>
  </storage>
```

```
</StorageListDocument>
```

### Create a storage

**POST /storage**

Creates a new storage.

**Accepts**

- **application/xml, application/json** – StorageDocument

**Produces**

- **application/xml, application/json** – StorageDocument
- **text/plain** – Storage id

**Role** _storage_write

---

**Note:** If your storage is to be used for autoimport, the `showImportables` property in the StorageDocument must be set to `true`. The files on the storage will then be discovered and available for use with `GET /storage/importable` or *GET /storage/auto-import/*.

---

### Example

```
POST /storage
Content-Type: application/xml

<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>LOCAL</type>
  <capacity>150000000000</capacity>
  <method>
    <uri>file:///mnt/ingest/</uri>
    <read>true</read>
    <write>true</write>
    <browse>true</browse>
  </method>
  <lowWatermarkPercentage>90</lowWatermarkPercentage>
  <highWatermarkPercentage>75</highWatermarkPercentage>
  <showImportables>true</showImportables>
</StorageDocument>
```

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-2</id>
  <state>NONE</state>
  <type>LOCAL</type>
  <capacity>150000000000</capacity>
  <freeCapacity>150000000000</freeCapacity>
  <method>
    <id>VX-4</id>
    <uri>file:///mnt/ingest/</uri>
    <read>true</read>
    <write>true</write>
    <browse>true</browse>
    <type>NONE</type>
  </method>
```

```
  <metadata/>
  <lowWatermark>112500000000</lowWatermark>
  <highWatermark>135000000000</highWatermark>
  <lowWatermarkPercentage>75</lowWatermarkPercentage>
  <highWatermarkPercentage>90</highWatermarkPercentage>
  <showImportables>true</showImportables>
</StorageDocument>
```

### Retrieve a storage

**GET /storage/** (*storage-id*)
Retrieves a specific storage.

**Produces**

- **application/xml, application/json** – StorageDocument

**Role** _storage_read

### Example

```
GET /storage/VX-2
```

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-2</id>
  <state>NONE</state>
  <type>LOCAL</type>
  <capacity>150000000000</capacity>
  <freeCapacity>150000000000</freeCapacity>
  <method>
    <id>VX-4</id>
    <uri>file:///mnt/ingest/</uri>
    <read>true</read>
    <write>true</write>
    <browse>true</browse>
    <type>NONE</type>
  </method>
  <metadata/>
  <lowWatermark>112500000000</lowWatermark>
  <highWatermark>135000000000</highWatermark>
  <lowWatermarkPercentage>75</lowWatermarkPercentage>
  <highWatermarkPercentage>90</highWatermarkPercentage>
  <showImportables>true</showImportables>
</StorageDocument>
```

### Update a storage

**PUT /storage/** (*storage-id*)
Updates an existing storage.

**Accepts**

- **application/xml, application/json** – StorageDocument

**Produces**

- **application/xml, application/json** – StorageDocument

- **text/plain** – Storage id

### Example

```
PUT /storage/VX-2
Content-Type: application/xml

<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <type>LOCAL</type>
  <capacity>150000000000</capacity>
  <lowWatermarkPercentage>80</lowWatermarkPercentage>
  <highWatermarkPercentage>95</highWatermarkPercentage>
</StorageDocument>
```

```
<StorageDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-2</id>
  <state>NONE</state>
  <type>LOCAL</type>
  <capacity>150000000000</capacity>
  <freeCapacity>150000000000</freeCapacity>
  <method>
    <id>VX-4</id>
    <uri>file:///mnt/ingest/</uri>
    <read>true</read>
    <write>true</write>
    <browse>true</browse>
    <type>NONE</type>
  </method>
  <metadata/>
  <lowWatermark>120000000000</lowWatermark>
  <highWatermark>142500000000</highWatermark>
  <lowWatermarkPercentage>80</lowWatermarkPercentage>
  <highWatermarkPercentage>95</highWatermarkPercentage>
  <showImportables>true</showImportables>
</StorageDocument>
```

### Delete a storage

**DELETE /storage/** (*storage-id*)

Deletes the storage. All files in storage will remain after call, but the Vidispine system will no longer manage them.

**Query Parameters**

- **safe** (*boolean*) –
  - `true` - Storage will only be deleted if there are no files connected to items on the storage.
  - `false` (default) - Storage will be deleted, and any file entities will be disconnected from items and deleted.

**Status Codes**

- **409 Conflict** – If `safe` parameter is `true` this error code will be given if there are files connected to items on the storage.

**Role** _storage_write

### Storage information

### Update the storage type

**PUT /storage/** (*storage-id*) **/type/**
*type* Sets the type of the storage.

**Role** _storage_write

### Retrieve the storage status

**GET /storage/** (*storage-id*) **/status**
Retrieves the status of the storage.

**Produces**

- **text/plain** – Status string

**Role** _storage_read

### Retrieve the amount of free space on a storage

**GET /storage/** (*storage-id*) **/freespace**
Retrieves the amount of free space on the storage.

**Produces**

- **text/plain** – Amount of free space as decimal number, between 0 and 100, inclusive

**Role** _storage_read

### Rescanning

### Rescan a storage

**POST /storage/** (*storage-id*) **/rescan**
Triggers a rescan of a single storage.

The `scanInterval` property can be used to control how often (in seconds) a storage is scanned. Default is `60`. By setting `scanInterval` to `-1` storage scans will be disabled. By calling `/rescan`, the system is forced to rescan a storage without delay.

**Query Parameters**

- **path** (*string*) – Rescans a specific path on storage to find importable files. This will not update deleted files.

  New in version 5.4.

### Storage methods

Storage methods specify the different ways of accessing files on a specific storage.

Credentials are encrypted. This means that passwords cannot be viewed through the API/server logs.

### List storage methods

**GET /storage/** (*storage-id*) **/method**
Retrieves the access methods configured on a specific storage.

**Query Parameters**

- **url** (*string*) – Only return methods with this URL. Wildcards (`?`, `*`) can be used, for example `http:*`.
- **read** (*boolean*) –
  - `true` - Only return methods which have file read capability.
  - `false` (default) - Return methods regardless of file read capability.
- **write** (*boolean*) –
  - `true` - Only return methods which have file write capability.
  - `false` (default) - Return methods regardless of file write capability.
- **browse** (*boolean*) –
  - `true` - Only return methods which have file browse capability.
  - `false` (default) - Return methods regardless of file browse capability.

**Produces**

- **application/xml, application/json** – StorageMethodListDocument

**Role** _storage_read

### Example

```
GET /storage/VX-2/method
```

```
<StorageMethodListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <method>
    <id>VX-4</id>
    <uri>file:///mnt/ingest/</uri>
    <bandwidth>0</bandwidth>
    <read>true</read>
    <write>true</write>
    <browse>true</browse>
    <lastSuccess>2014-07-04T09:55:09.779+02:00</lastSuccess>
    <type>NONE</type>
  </method>
</StorageMethodListDocument>
```

### Update or create a storage method

**PUT /storage/** (*storage-id*) **/method**

Adds a new access method to the storage. If URL matches an existing method, a new method is not created, instead the existing one is updated.

**Query Parameters**

- **url** (*string*) – Required. Method is accessed through this URL
- **read** (*boolean*) –
  - `true` (default) - Method has file read capability
  - `false` - Method does not have file read capability
- **write** (*boolean*) –
  - `true` (default) - Method has file write capability

- `false` - Method does not have file write capability

- **browse** (*boolean*) –
  - `true` (default) - Method has file browse capability
  - `false` - Method does not have file browse capability
- **bandwidth** (*integer*) – The *bandwidth* of this method in bytes per second. Default `0`.
- **type** (*string*) – *method type*. Default is empty.

**Produces**

- **text/plain** – *Tabbed tuples* : id, URL, status string of method

**Role** _storage_write

### Example

```
PUT /storage/VX-2/method?url=http://10.12.0.3/ingest/&read=true&write=false&
↪browse=false
```

```
VX-5        http://10.12.0.3/ingest/        NONE
```

### Retrieve a storage method

**GET /storage/** (*storage-id*) **/method/**
*method-id* Retrieves a specific access method to storage.

**Produces**

- **application/xml, application/json** – StorageMethodListDocument

**Role** _storage_read

### Create/update a storage method

**PUT /storage/** (*storage-id*) **/method/**
*method-id* Updates access method to the storage.

**Query Parameters**

- **url** (*string*) – Required. Method is accessed through this URL
- **read** (*boolean*) –
  - `true` - Method has file read capability.
  - `false` - Method does not have file read capability.
- **write** (*boolean*) –
  - `true` - Method has file write capability.
  - `false` - Method does not have file write capability.
- **browse** (*boolean*) –
  - `true` - Method has file browse capability.
  - `false` - Method does not have file browse capability.
- **bandwidth** (*integer*) – The *bandwidth* of this method in bytes per second. Default `0`.

- **type** (*string*) – *method type*.

**Produces**

- **text/plain** – *Tabbed tuples* : id, URL, status string of method

**Role** _storage_write

### Delete a storage method

**DELETE /storage/** (*storage-id*) **/method/** *method-id*
**DELETE /storage/** (*storage-id*) **/method**
Removes an access method from a storage.

**Query Parameters**

- **url** (*string*) – Method is accessed through this URL

**Role** _storage_write

### Importing/exporting a storage definition

Vidispine can export a definition document describing all the files within a storage, which can later be imported to a new storage on a different Vidispine instance.

### Export a storage definition

**GET /storage/** (*storage-id*) **/export**
Creates a StorageImportDocument that describes every file on the storage. This should be saved to a file which can later be used to import the storage definition.

**Produces**

- **application/xml, application/json** – StorageImportDocument

**Role** _storage_read

### Import a storage definition

**POST /storage/import**
Creates a new storage based on the StorageImportDocument. A file entity will be created for each entry in the document, if a file with that ID does not already exist. Finally, a storage method will be added, with the path supplied in the call.

**Query Parameters**

- **path** (*string*) – The file system path to where the files are located.

**Accepts**

- **application/xml, application/json** – A StorageImportDocument

**Role** _storage_write

### Evacuating storages

If you would like to delete a storage, but you still have files there which are connected to items, you can first trigger an evacuation of the storage. This will cause Vidispine to attempt to delete redundant files, or move files to other storages. Once the evacuation is complete, the storage will get the state `EVACUATED`.

### Evacuate a storage

**PUT /storage/** (*storage-id*) **/evacuate**
Trigger evacuation of a storage.

**Role** _storage_write

### Cancel evacuation of a storage

**DELETE /storage/** (*storage-id*) **/evacuate**
Cancel the evacuation process on a storage.

**Role** _storage_write

### Key-value metadata

Storages support *key-value metadata*.

### Reserved keys

Certain keys are used to control the behavior of a storage, they must not be used to store generic metadata.

**closeLimit**
An integer specifying the number of minutes until the system automatically considers an open file to be closed.

**Default** `5`

**lostLimit**
An integer specifying the number of minutes until the system automatically considers a missing file to be lost.

**Default** `10`

**toAppearLimit**
An integer specifying the number of minutes the system waits for a file to appear before considering it to be missing.

**Default** `10`

**refreshInterval**
The interval (in seconds) that Vidispine will compare the list of file seen on disk with the list of files in the database. This to fix any possible mismatch.

**Default** `900`

**hashMode**
Set to `false` to disable hash computation for this storage.

**Default** `true`

**additionalHash**
A comma-separated list of additional hashing algorithms that should be applied to files. E.g.: `MD5,SHA-256`.

The hash algorithm `FILENAME` is also supported, which takes the last part of the path concatenated with the file size, and calculates a SHA-1 checksum of that string.

**Default** (none)

**fileListBatchSize**
Maximum size of list of files that are updated in the database at the same time.

**Default** `100000`

**keepMissingFiles**
If set to `false` then missing files that do *not* belong to any items will be removed from the database instead of being marked as lost.

**Default** To the value of the configuration property *keepMissingFiles*.

**keepEmptyDirectories**
Do not delete empty parent directories when deleting the last file in a directory, see *Parent directory management*.

**Default** To the value of the configuration property *keepEmptyDirectories*.

**statsPerSecond**
An integer specifying the maximum number of `stat` system calls that are made to the storage (only `file://` URIs). See also the configuration property *statsPerSecond*.

**Default** (none)

**scanOnStart**
By default, Vidispine will scan the file system on start-up. Storages that should never be scanned, that should solely use SQS or SNS for example, should have this parameter set to `false`.

**Default** `true`

**Since** 5.0

**refreshOnStart**
By default, Vidispine synchronizes the database with the file system on start-up. On very large systems, this can take a very long time. Set this parameter to `false`, and use the rescan functionality (see below) instead.

**Default** `true`

**deleteFileIfNotFound**
If set to `true`: if file that is marked for deletion cannot be deleted, but is not found on the file system, it is assumed to be deleted.

**Default** `false`

**deleteFileIfReadOnly**
If a file that is marked for deletion cannot be deleted, and the file exists on a read-only storage, then:

- If set to `true`: delete the file from the database, as if the file had been deleted.
- If set to a *file state*: set the file to this state.

**Default** `false`

**Example** `true`, `ARCHIVED`, `LOST`

**excludeFilter**
A regular expression. Any file with a path (relative to the storage's root folder) that matches the expression will be ignored. Note that the expression must match the entire path, not only a part of the path.

**Default** (none)

**detectRenamedFiles**
If this metadata is set to `true`, whenever a file belonging to an item is marked as `LOST`, and there is another file on the same storage with the same hash value, the new file is associated with the item instead.

Other possible values are `all`, where the old (lost) file resided on any storage, or a comma-separated list of identifiers of storages on which the old (lost) file resided on.

**Default** (none)

**probeFileBeforeClosing**
If this metadata is set to `true`, Vidispine will ensure that `OPEN` files are readable and are not being modified by any other process before marking them as `CLOSED`. Jobs with transfer steps will be set to `WAITING` with a `FileUnavailable` *job problem* until the file becomes available. This is useful for Windows storages where the size and metadata is constant while a file is copied to the storage, but the file cannot be read.

**Default** `false`

**vxaId**
If this metadata field is set, the storage will be automatically deleted if the share is removed from the *Vidispine Server Agent*.

**Default** (none) for regular storages, automatically set by VSA when a share is added and a storage is created.

**sqsName**
The name of Amazon SQS queue to poll for S3 file events. See *S3 Event SQS Notifications* for more information.

This key is only read for S3 methods.

**sqsEndpoint**
The endpoint of Amazon SQS queue.

This key is only read for S3 methods.

Example: `sqs.eu-west-1.amazonaws.com`

**snsTopic**
The ARN of an SNS topic to receive S3 file events from. See *S3 Event SNS Notifications* for more information.

**Example** `arn:aws:sns:eu-west-1:123456791011:topic_name`

**s3EventETag**
When receiving file events from SNS this can be specified to set the ETag included in the message from the storage as metadata on the file.

**Example** `true`

**Default** `false`

**s3EventTime**
When receiving file events from SNS this can be specified to set the timestamp in the message received as metadata on the file.

**Example** `true`

**Default** `false`

**hashingThreadCount**
The number of hashing thread that should be run on the VSA side. Note: this configuration only work for VSA storage.

**Default** `1`

**noDefaultHash**
Do not calculate hash checksums for files on the storage.

Example: `true`

**Default** `false`

**verifyHashAfterTransfer**
If a job is copying or moving a file to this storage the hash of the source file and the hash of the newly copied/moved file are compared to ensure that the new file is binary identical to the original file. If the hashes differ the job fails and the destination file is removed.

- If set to `true`: hash verification is performed after the file has arrived on this storage.
- If set to `false`: no hash verification is performed.

The hash verification procedure makes use of the hash calculation that is happening on all storages by default (see *File hashing*). If hash calculation is disabled on the source storage the hash is calculated *before* the copy/move operation is performed. If hash calculation is disabled on the destination storage the hash is calculated *after* the copy/move operation has completed. These hashes are stored in the Vidispine database for later reuse.

Note that hash calculation requires reading the whole file and thus is a lengthy process that can significantly increase execution time of copy/move jobs.

**Default** `false`

**`storageActivationFile`**
Require a .storage file to be present in the storage method's URI for storage to register as online. Takes precedence over the *`global setting`*.

**Default** (none) - The global setting applies.

**Since** 4.17

**`scanMethodAlgorithm`**
Specify which scan algorithm to use, see *Storage scanning algorithm*.

**Default** To the value of the configuration property *`scanMethodAlgorithm`*.

**Since** 5.5.2

### Size units

In the query parameters above, size can be specified in number of bytes as `integer [ [ whitespace ] unit ]` where the multiplier unit can be one of (case insensitive):

| Unit | Factor |
|---|---|
| `KB` | $1000^1$ |
| `K`, `KiB` | $2^{10}$ |
| `MB` | $1000^2$ |
| `M`, `MiB` | $2^{20}$ |
| `GB` | $1000^3$ |
| `G`, `GiB` | $2^{30}$ |
| `TB` | $1000^4$ |
| `T`, `TiB` | $2^{40}$ |
| `PB` | $1000^5$ |
| `P`, `PiB` | $2^{50}$ |
| `EB` | $1000^6$ |
| `E`, `EiB` | $2^{60}$ |
| `ZB` | $1000^7$ |
| `Z`, `ZiB` | $2^{70}$ |
| `YB` | $1000^8$ |
| `Y`, `YiB` | $2^{80}$ |

It should be noted that currently, Vidispine has a 64-bit limit on the size of a storage, which means that multipliers larger than `ZB` are of limited use.

## 17.28.4 Storage groups

Storages can be organized in zero or more storage groups. Use storage groups to:

- Allow users to group storages in your applications.
- Apply storage rules to these groups.

### Storage group metadata

Storages groups support *key value metadata*.

### Managing storage groups

### List all storage groups

**GET /storage/storage-group/**
Retrieves all storage groups known by the system.

**Produces**

- **application/xml, application/json** – StorageGroupListDocument

**Role** _storage_group_read

### Create a storage group

**PUT /storage/storage-group/** (*group-name*)
Creates a new storage group with the specified name. If the group already exists this operation does nothing.

**Status Codes**

- **200 OK** – Group created successfully.

**Role** _storage_group_write

### Delete a storage group

**DELETE /storage/storage-group/** (*group-name*)
Removes the storage group with the given name. Note that this operation does not remove the actual storages contained in the group.

**Status Codes**

- **200 OK** – Group removed successfully.
- **404 Not found** – No group with that name exists.

**Role** _storage_group_write

### Storage group content

### List all storages within a group

**GET /storage/storage-group/** (*group-name*)
Lists all storages belonging to a certain group.

**Status Codes**

- **404 Not found** – No group with that name exists.

**Produces**

- **application/xml, application/json** – StorageGroupDocument

**Role** _storage_group_read

#### Add a storage to a group

**PUT /storage/storage-group/**(*group-name*)**/**
*storage-id* Adds a storage to a group. If that group already contains the specified storage this operation does nothing.

**Status Codes**

- **200 OK** – Group added successfully.
- **404 Not found** – No group with that name or no storage with that id exists.

**Role** _storage_group_write

#### Remove a storage from a group

**DELETE /storage/storage-group/**(*group-name*)**/**
*storage-id* Removes a storage from a group.

**Status Codes**

- **200 OK** – Group removed successfully.
- **404 Not found** – No group with that name exists, or the group does not contain that storage.

**Role** _storage_group_write

### 17.28.5 Storage name rules

Storage name rules can be used to enforce a specific naming scheme for files created on a specific storage.

#### Working with storage name rules

#### List all storage name rules applied on a shape

**GET /item/**(*item-id*)**/shape/**
*shape-id***/filename** Retrieves a list of URIs to all storage name rules that are contained within a shape.

**Produces**

- **application/xml, application/json** – URIListDocument
- **text/plain** – CRLF-delimited list of URIs

**Role** _storage_rule_read

#### Create a new storage name rule

**PUT /item/**(*item-id*)**/shape/**
*shape-id***/filename/***storage-id* Creates a new storage name rule that attempts enforce the filename on a certain storage. This operation does not rename the file, it merely creates a rule for it. The file will then be renamed at a later time, if the file is located on that storage.

**Query Parameters**

- **filename** (*string*) – Required. The desired filename of the file.

**Status Codes**

- **200 OK** – Rule added successfully.
- **400 Bad request** – A conflicting rule exists.

**Role** _storage_rule_write

#### Delete a storage name rule

**DELETE /item/** (*item-id*) **/shape/** *shape-id***/filename/***storage-id* Deletes a storage name rule that matches the (item id, shape id, storage id, filename) quadruple. Note that this does not change any existing filenames a file might have.

**Query Parameters**

- **filename** (*string*) – Required. The filename of the rule.

**Status Codes**

- **200 OK** – Rule removed successfully.

**Role** _storage_rule_write

### 17.28.6 Storage rules

#### Creating/modifying/reading storage rules

Storage rules can be applies on entities within the item, collection, library and shape tag resources. Note that for the shape tag resource no default rule can be set.

In the following reference, `{rule-resource}` is one of the following:

- `/item/{item-id}/storage-rule`
- `/collection/{collection-id}/storage-rule`
- `/library/{library-id}/storage-rule`
- `/shape-tag/storage-rule`

#### List all storage rules

**GET /storage-rule/**
Retrieves all storage-rules that matches the given parameters.

**Query Parameters**

- **type** (*string*) – Comma-separated list of types to retrieve, if not specified all types will be retrieved. Valid values are `ITEM`, `COLLECTION`, `LIBRARY` and `GENERIC`.
- **tag** (*string*) – Comma-separated list of tags to retrieve, if not specified rules of all tag types will be retrieved.

**Produces**

- **application/xml, application/json** – StorageRulesDocument

**Role** _storage_rule_read

#### Example

```
GET /storage-rule?type=LIBRARY,COLLECTION&tag=original,lowres
```

```
<StorageRulesDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <tag id="lowres">
      <storageCount>1</storageCount>
      <storage>VX-1</storage>
```

```
      <appliesTo>
         <id>VX-20</id>
         <type>COLLECTION</type>
      </appliesTo>
      <precedence>HIGH</precedence>
   </tag>
   <tag id="original">
      <storageCount>2</storageCount>
      <appliesTo>
         <id>*68</id>
         <type>LIBRARY</type>
      </appliesTo>
      <precedence>MEDIUM</precedence>
   </tag>
</StorageRulesDocument>
```

### List all storage rules that applies to a certain shape

**GET /item/** (*item-id*) **/shape/** *shape-id***/storage-rule** Retrieves the storage rules that applies to a certain shape in a sorted manner. The rules are sorted according to priority, with the most important rule being first and the least important rule being last.

**Query Parameters**

- **all** (*boolean*) –
  - `true` - Return all rules, regardless whether another rule overwrites it or not.
  - `false` (default) - Return only rules that are in effect.

**Produces**

- **application/xml, application/json** – StorageRulesDocument

**Role** _storage_rule_read

### List all storage rules for an entity

**GET {rule-resource}**
Retrieves all storage rules that are applied on a certain entity in a certain resource.

**Produces**

- **application/xml, application/json** – StorageRulesDocument

**Role** _storage_rule_read

### Example

```
GET /storage-rule/
```

```
<StorageRulesDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <default>
    <storageCount>2</storageCount>
    <priority level="1">capacity</priority>
    <priority level="2">bandwidth</priority>
    <storage>VX-122</storage>
  </default>
```

```
  <tag id="lowres">
    <storageCount>3</storageCount>
    <storage>VX-123</storage>
  </tag>
  <tag id="web">
    <priority level="1">bandwidth</priority>
    <priority level="2">capacity</priority>
    <storage>VX-124</storage>
  </tag>
</StorageRulesDocument>
```

### Set a default rule

**PUT {rule-resource}**
Sets the default rule.

**Status Codes**

- **200 OK** – Rule set successfully.
- **400 Bad request** – The request was malformed.
- **404 Not found** – Could not find a specified storage.

**Accepts**

- **application/xml, application/json** – StorageRuleDocument

**Role** _storage_rule_write

### Example

```
PUT /collection/VX-45/storage-rule
Content-Type: application/xml

<StorageRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <storageCount>2</storageCount>
  <priority level="1">capacity</priority>
  <priority level="2">bandwidth</priority>
  <storage>VX-122</storage>
</StorageRuleDocument>
```

### Delete a default rule

**DELETE {rule-resource}**
Deletes the default rule.

**Status Codes**

- **200 OK** – Rule deleted successfully.
- **404 Not found** – Could not find a default rule.

**Role** _storage_rule_write

### Example

```
DELETE /collection/VX-45/storage-rule
```

```
200 OK
```

### Retrieve a a storage rule

**GET {rule-resource}/** (*tag-name*)
Returns the rule that is applied to a certain shape tag.

**Status Codes**

- **404 Not found** – No shape tag with that that name could be found.

**Produces**

- **application/xml, application/json** – StorageRuleDocument

**Role** _storage_rule_read

### Example

```
GET /collection/VX-45/storage-rule/lowres
```

```
<StorageRuleDocument id="lowres" xmlns="http://xml.vidispine.com/schema/vidispine">
   <storageCount>3</storageCount>
   <priority level="1">capacity</priority>
   <priority level="2">bandwidth</priority>
   <storage>VX-123</storage>
</StorageRuleDocument>
```

### Set a storage rule

**PUT {rule-resource}/** (*tag-name*)
Updates a storage rule applied to a certain shape tag.

**Status Codes**

- **200 OK** – Rule set successfully.
- **400 Bad request** – The request was malformed.
- **404 Not found** – Could not find a specified storage or a shape tag with that name.

**Accepts**

- **application/xml, application/json** – StorageRuleDocument

**Role** _storage_rule_write

### Example

```
PUT /collection/VX-45/storage-rule/lowres
Content-Type: application/xml

<StorageRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <storageCount>3</storageCount>
  <storage>VX-123</storage>
</StorageRuleDocument>
```

#### Delete a storage rule

**DELETE {rule-resource}/**(*tag-name*)
Deletes a specific rule.

**Status Codes**

- **200 OK** – Rule deleted successfully.
- **404 Not found** – Could not find a shape tag with that name or a specific rule applied to that tag.

**Role** _storage_rule_write

## 17.29 Task definitions

Jobs are made up of a number of tasks that execute in a specific order.

### 17.29.1 Task definitions

#### Retrieve task definitions

**GET /task-definition**
Retrieves all tasks that have been defined in the system.

**Query Parameters**

- **type** (*string*) – *Job type* to retrieve task definitions for.

**Produces**

- **application/xml, application/json** – TaskDefinitionListDocument

**Role** _taskdefinition_read

#### Retrieve task definitions by type

**GET /task-definition/jobtype/**(*type*)
Retrieves the tasks that have been defined for a specific job type.

**Produces**

- **application/xml, application/json** – TaskDefinitionListDocument

**Role** _taskdefinition_read

#### Define new task

**POST /task-definition**
Defines one or more new tasks.

**Query Parameters**

- **url** (*boolean*) –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.

**Accepts**

- **application/xml, application/json** – TaskDefinitionListDocument

**Produces**

- **application/xml, application/json** – URIListDocument

**Role** _taskdefinition_write

### Retrieve a task

**GET /task-definition/** (*task-id*)
**GET /task-definition/jobtype/** (*type*) **/step/**
*step* Retrieves the definition document for a task with a specific id.

**Produces**

- **application/xml, application/json** – TaskDefinitionDocument

**Role** _taskdefinition_read

### Validate a task

**GET /task-definition/** (*task-id*) **/validate**
**GET /task-definition/jobtype/** (*type*) **/step/**
*step***/validate** Verifies that the bean referred to in the task can be resolved and that it contains the specified method.

Does nothing if the task is a script task.

**Status Codes**

- **200** – The bean and method exists.
- **400** – If the bean or method could not be found.

**Produces**

- **text/plain** – Informational status text.

**Role** _taskdefinition_read

### Delete a task

**DELETE /task-definition/** (*task-id*)
**DELETE /task-definition/jobtype/** (*type*) **/step/**
*step* Deletes the task.

**Role** _taskdefinition_write

### Update an existing task

**PUT /task-definition/** (*task-id*)
**PUT /task-definition/jobtype/** (*type*) **/step/**
*step* Updates the task.

**Accepts**

- **application/xml, application/json** – TaskDefinitionDocument

**Produces**

- **application/xml, application/json** – TaskDefinitionDocument

**Role** _taskdefinition_write

## 17.29.2 Custom job types

### Create a new custom job type

**POST /task-definition/jobtype/** (*type*)

Creates a new job type with the specified name. The recommended format of the `type` path parameter is `{VENDOR_PREFIX}_{JOB_TYPE}`.

**Query Parameters**

- **id** (*integer*) – Required. An integer between 20000 and 30000, must be unique among job types.

**Status Codes**

- **409** – Name or id already taken.
- **400** – If name or id was not specified.

**Produces**

- **application/xml, application/json** – TaskDefinitionListDocument

**Role** _taskdefinition_write

### Delete a custom job type

**DELETE /task-definition/jobtype/** (*type*)

Deletes the job type with the specified name. This will only work for custom job types. System defined job types cannot be deleted.

**Status Codes**

- **404** – Custom job type with specified name not found.

**Role** _taskdefinition_write

## 17.29.3 Task definition scripts

### Retrieve the script for a task definition

**GET /task-definition/** (*task-id*) **/script**
**GET /task-definition/jobtype/** (*type*) **/step/** *step***/script** Retrieves the script of the task definition.

**Produces**

- **application/javascript** – A JavaScript

**Status Codes**

- **204** – No script is set for the task definition.

**Role** _taskdefinition_read

### Set a script for a shape tag

**PUT /task-definition/** (*task-id*) **/script**
**PUT /task-definition/jobtype/** (*type*) **/step/** *step***/script** Sets a script for the task definition.

**Accepts**

- **application/javascript** – A JavaScript

**Role** _taskdefinition_write

### 17.29.4 Job graphs

In order to easily see the dependencies between steps for a particular job type, there is functionality to render the job definition as a graph. In order to render the graph, the Graphviz (http://www.graphviz.org/) package is required.

#### Get job graph

**GET /task-definition/jobtype/**(*type*)**/graph**
Shows the dependencies of the tasks of a specified *job type*.

**Produces**

- **image/png** –

**Role** _administrator

#### Get job graph as DOT file

**GET /task-definition/jobtype/**(*type*)**/graph/dot**
Shows the dependencies of the tasks of a specified *job type* in DOT format, for further processing.

**Produces**

- **text/plain, text/vnd.graphviz** –

**Role** _administrator

## 17.30 Task groups

Task groups can be used to control the resources that a specific job should use.

### 17.30.1 Task groups

#### List all task groups

**GET /task-group**
Retrieves all task groups that have been defined in the system.

**Produces**

- **application/xml, application/json** – TaskGroupListDocument

**Role** _administrator

#### Update or create a task group

**PUT /task-group/**(*group-name*)
Updates or creates the task group with the given name.

**Accepts**

- **application/xml, application/json** – TaskGroupDocument

**Produces**

- **application/xml, application/json** – TaskGroupDocument

**Role** _administrator

### Example

```
PUT /task-group/imports HTTP/1.1
Accept: application/xml
Content-Type: application/xml

<TaskGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <job>
    <type>PLACEHOLDER_IMPORT</type>
  </job>
  <transcoder>
    <id>VX-1</id>
  </transcoder>
</TaskGroupDocument>
```

```
<TaskGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <name>imports</name>
  <job>
    <type>PLACEHOLDER_IMPORT</type>
  </job>
  <transcoder>
    <id>VX-1</id>
  </transcoder>
</TaskGroupDocument>
```

### Create or update a task group

**PUT /task-group/** (*group-name*)
Creates or updates the task group with the given name.

This request is identical to the above, except that the task group parameters are specified using query parameters instead of using a TaskGroupDocument.

**Query Parameters**

- **priority** (*string*) – The priority of the task group.
- **transcoder** (*string[]*) – The id of the transcoder(s) to include in the group.
- **job** (*string[]*) – The job criteria(s). One of:
  - `type:{type}` - Jobs of this type.
  - `priority:{priority}` - Jobs with this priority.
  - `user:{username}` - Jobs created by this user.
  - `group:{groupname}` - Jobs created by a user in this group.
  - `data:{key}[={value}]` - Jobs with this job data.
- **metadata** (*string[]*) – Any key-value metadata to set on the group. Format is `{key}={value}`.

**Produces**

- **application/xml, application/json** – TaskGroupDocument

**Role** _administrator

#### Example

```
PUT /task-group/imports?job=type:PLACEHOLDER_IMPORT&transcoder=VX-1 HTTP/1.1
Accept: application/xml
```

```
<TaskGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <name>imports</name>
  <job>
    <type>PLACEHOLDER_IMPORT</type>
  </job>
  <transcoder>
    <id>VX-1</id>
  </transcoder>
</TaskGroupDocument>
```

### Retrieve a task group

**GET /task-group/** (*group-name*)
Retrieves the task group document for a task group with a specific name.

**Produces**

- **application/xml, application/json** – TaskGroupDocument

**Role** _administrator

### Delete a task group

**DELETE /task-group/** (*group-name*)
Deletes the task group.

Note that this only deletes the task group. It will not remove any transcoders or other resources that are in the group.

**Role** _administrator

### Delete all task groups

**DELETE /task-group**
Deletes all task groups.

Note that this only deletes the task groups. It will not remove any transcoders or other resources that are in the groups.

**Role** _administrator

## 17.30.2 Task group transcoders

### Add a transcoder to a task group

**PUT /task-group/** (*group-name*) **/transcoder/**
*transcoder-id* Adds the transcoder to the specified task group.

**Role** _administrator

### Remove a transcoder from a task group

**DELETE /task-group/** (*group-name*) **/transcoder/**
*transcoder-id* Removes a transcoder from the specified task group.

**Role** _administrator

### 17.30.3 Key-value metadata

Task groups support *key-value metadata*.

## 17.31 Transfers

List active transfers. A transfer is normally started while doing a *import using the request body*.

### 17.31.1 Overview

#### Transfer state

A transfer can be in one of the following states.

**TRANSFERRING** Data is currently being sent.

**WAITING** The transfer is waiting to start.

**FINISHED** All the data has been transferred.

**ABORTED** The transfer has been aborted by the user.

**FAILED** An error has occurred causing the transfer to fail.

**FINISHED_PART** A piece of the data has been sent.

#### Priorities

Transfers are assigned bandwidth according to their priorities. A priority is an integer between 1 and 1000. Transfers with a higher priority value is prioritized over transfers with a lower priority value.

### 17.31.2 Managing transfers

#### List all transfers

**GET /transfer**
Returns all transfers that are in a particular state.

**Query Parameters**

- **state** (*string*) – *Transfer state* of the transfers to retrieve. Default is `TRANSFERRING`.
- **number** (*integer*) – The number of transfers to return. Default is all transfers.
- **first** (*integer*) – The number of the first transfer to return. Default is `1`.

**Produces**

- **application/xml, application/json** – A TransferListDocument
- **text/plain** – CRLF-delimited list of transfer ids

**Role** _transfer_read

#### Retrieve a transfer

**GET /transfer/** (*transfer-id*)
Retrieves a specific transfer.

**Produces**

- **application/xml, application/json** – A TransferDocument

**Role** _transfer_read

#### Update the priority of a transfer

**PUT /transfer/** (*transfer-id*)
Sets a new priority for a specific transfer.

**Query Parameters**

- **transferPriority** (*integer*) – Required. The desired priority.

**Role** _transfer_write

## 17.32 Transfer log

The transfer log records low-level file transfers. It is typically used for troubleshooting, to be able to determine what happened when.

### 17.32.1 Examining the log

#### List all transfer log entries

**GET /log/transfer-log**
Retrieves log entries according to the specified filtering criteria. Note that the transfer log table does not have a lot of indices, so the extraction of data can be slow. Do not use this method other than for troubleshooting.

**Query Parameters**

- **first** (*integer*) – Number of first row to return. Default is `0`.
- **rows** (*integer*) – Number of rows to return. Default is `100`. Cannot be greater than 1000.
- **starttime** (*string*) – ISO 8601 time, for lower limit of rows to return.
- **endtime** (*string*) – ISO 8601 time, for upper limit of rows to return.
- **storage** (*string*) – Site id, only return transfers where source or destination storage matches. Default is all rows. Note that not all transfers contains information about storages.
- **file** (*string*) – Site id, only return transfers where source or destination file matches. Default is all rows. Note that not all transfers contains information about files.
- **item** (*string*) – Site id, only return transfers where source or destination item matches. Default is all rows. Note that not all transfers contains information about items.
- **shape** (*string*) – Site id, only return transfers where source or destination shape matches. Default is all rows. Note that not all transfers contains information about shapes.
- **uri** (*string*) – URI, only return transfers where source or destination URI matches. Default is all rows. A star (`*`) in the URI represents a wildcard.
- **job** (*string*) – Site id, only return transfers where job matches. Default is all rows. Note that not all transfers contains information about jobs.
- **status** (*string*) – Comma-separated list. Only return transfers where status matches. Valid values are `NONE`, `STARTING`, `FINISHED`, `FAILED`.
- **method** (*string*) – Comma-separated list Only return transfers where method matches. Valid values are `GET`, `POST`, `PUT`, `DELETE`.

- **performCount** (*boolean*) –
  - `true` - Return a total number of rows matching criteria (except first and count).
  - `false` (default) - Do not return a total number of rows matching criteria.

**Produces**

- **application/xml, application/json** – TransferLogListDocument

**Role** _administrator

# 17.33 Users

## 17.33.1 Users

Manage users.

### Managing users

### List all users

**GET /user**

Retrieves a list of all known users.

**Query Parameters**

- **name** (*string[]*) – List of user names to return information about. Default is all users.
- **disabled** (*string*) – If `true` only disabled users are shown, if `false` only enabled users are shown. Default is `all` - all users are shown.
- **first** (*integer*) – Start returning users from specified number. Default is `1`, the beginning of the list.
- **number** (*integer*) – Return at most specified number of users. Default is no limit.

**Produces**

- **application/xml, application/json** – UserListDocument

**Role** _user_read

### Create a user

**POST /user**

Creates a new user based on the information in the UserDocument.

**Query Parameters**

- **passwordType** (*string*) –
  - `raw` - Password is in plaintext.
  - `md5` (default) - Password is already hashed.

**Status Codes**

- **200 OK** – User created.
- **409 Conflict** – A user with that username already exists.

**Accepts**

- **application/xml, application/json** – UserDocument

**Role** _administrator

### Example

```
POST /user?passwordType=raw
Content-Type: application/xml

<UserDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <userName>myuser</userName>
  <realName>My User</realName>
  <password>qwerty</password>
  <groupList>
    <group>
      <groupName>mygroup</groupName>
    </group>
  </groupList>
</UserDocument>
```

### Retrieve a user

**GET /user/** (*username*)
Returns a specific user.

**Produces**

- **application/xml, application/json** – XML/JSON, schema UserDocument

**Role** _user_read

### Create a user

**PUT /user/** (*username*)
Creates a new user with the given username.

**Status Codes**

- **200 OK** – User created.
- **409 Conflict** – A user with that username already exists.

**Role** _administrator

### Update a user

**PUT /user/** (*username*)
Updates a user based on the information in the UserDocument.

The username of a user can be changed by providing a different username in the given user document.

**Query Parameters**

- **passwordType** (*string*) –
  - raw - Password is in plaintext.
  - md5 (default) - Password is already hashed.

**Status Codes**

- **200 OK** – User updated.
- **409 Conflict** – A user with the new username already exists.

**Accepts**

- **application/xml, application/json** – UserDocument

**Produces**

- **application/xml, application/json** – UserDocument

**Role** _administrator

**Caution:** To reflect a username change in the search index a re-index operation on items and collections is required. See *Re-indexing metadata*.

### Example

For example, to change the name and username of a user:

```
PUT /user/stephen@example.com
Content-Type: application/xml

<UserDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <userName>stephen</userName>
  <realName>Stephen</realName>
</UserDocument>
```

### Disable a user

**DELETE /user/** (*username*)

Disables a user with the given username, rendering that user unable to login.

**Query Parameters**

- **hard** (*boolean*) – If set to `true`, the user will be removed completely, including all access controls granted by the user. Else the user will be disabled. Default is `false`.
- **preserveAccess** (*boolean*) – If set to `true`, the access granted by the user will still apply. Only applicable for `hard=false`. Default is `false`.

  New in version 4.17.
- **transferAccess** (*string*) – If set, the specified user assumes all access controls of the user being deleted, including ownership, granted, and received access. This avoids any access chains breaking. The modified access control entries can be found in the *job document metadata*, under the `transferredAccess` key. Only applicable for `hard=true`.

  New in version 5.3.
- **notification** (*string*) – The *placeholder job notification* to use for this job.
- **notificationData** (*string*) – Any additional data to include for *notifications* on this job.
- **priority** (*string*) – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** (*string[]*) – Additional *information* for the job task.

**Produces**

- **application/xml, application/json** – XML/JSON, schema JobDocument if `hard=true`

**Role** _administrator

### Re-enable a user

**PUT /user/** (*username*) **/enable**
Re-enables a user with the given username, that has previously been disabled.

**Role** _administrator

### Search users

**PUT /user**
Simple search of fields `username`, `realname`, `groupname` and metadata.

Changed in version 5.1: Support for searching using `groupname` was added.

**Query Parameters**

- **number** (*integer*) – Return at most specified number of users. Default is `10`.
- **first** (*integer*) – Start returning users from specified number. Default is `1`, the beginning of the list.
- **disabled** (*string*) – If `true` only disabled users are shown, if `false` only enabled users are shown. Default is `all` - all users are shown.
- **ignoreCase** (*boolean*) –
  - `true` - Perform a case insensitive search on the fields username, realname, groupname and metadata values.
  - `false` (default) - Perform a case sensitive search.

  New in version 5.2.

**Accepts**

- **application/xml, application/json** – UserSearchDocument

**Produces**

- **application/xml, application/json** – UserListDocument
- **text/plain** – CRLF-delimited list of *Tabbed tuples*: `username`, `realname`, `groups*`

**Role** _user_read

### Example

```
PUT /user
Content-Type: application/xml

<UserSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <field>
        <name>username</name>
        <value>vidi</value>
    </field>
    <field>
        <name>key</name>
        <value>value</value>
    </field>
</UserSearchDocument>
```

Note that keywords `username`, `realname`, `disabled`, `groupname`, `preserveaccess`, `external`, `uuid`, and `origin` are reserved for searches on the respective user properties.

The boolean operators `AND`, `NOT`, and `OR` are supported.

```
<UserSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <field>
        <name>username</name>
        <value>vidi</value>
    </field>
    <field>
        <name>realname</name>
        <value>vidispine</value>
    </field>
    <operator operation="OR">
        <field>
          <name>key1</name>
          <value>value1</value>
        </field>
        <field>
          <name>key2</name>
          <value>value2</value>
        </field>
    </operator>
</UserSearchDocument>
```

Note when searching for groupnames the result is filtered on matching groupnames, and when using NOT the rest of the search criterias is handled similare to an OR search.

### User information

### Key-value metadata

Users support *key-value metadata*.

Users can always read their own metadata, but the `_user_metadata_write` role is required to edit it. The `_administrator` role is required to read or write other user's metadata.

### Retrieve the real name of a user

**GET /user/** (*username*) **/realname**
Returns the real name of a user.

**Produces**

- **text/plain** – The real name of the user.

**Role** _user_read

### See access control entries for user

New in version 5.4.

**GET /user/** (*username*) **/access**
Returns an AccessControlListDocument of all access entries that apply for the specified user. Entity type and entity id can be found in the `loc` field of the AccessControlDocument.

**Query Parameters**

- **entityType** (*string*) – The type of entity the access control applies to. One of `item`, `collection`, `library`, or `all`. Default is `all` - all entries are shown.
- **level** (*string*) – The *Access levels* that the access control grants. Default is all entries shown.
- **number** (*integer*) – Return at most specified number of entries. Default is all access controls.
- **first** (*integer*) – Start returning entries from specified number. Default is `1`, the beginning of the list.

**Produces**

- **application/xml, application/json** – AccessControlListDocument

**Role** _user_read (except for reading own access), _accesscontrol_read

### Update the real name of a user

**PUT /user/** (*username*) **/realname**
Changes the real name of a user.

**Accepts**

- **text/plain** – The new name.

**Role** _administrator

### User credentials

### Update the password of a user

**PUT /user/** (*username*) **/password**
Changes the password for a user. Hashed passwords are assumed to be represented as hexadecimal strings.

Any hashed passwords need to be salted using the *salt of the user*.

**Query Parameters**

- **type** (*string*) –
  - `raw` - Password is in plaintext.
  - `md5` (default) - Password is already hashed.

**Accepts**

- **text/plain** – The new password.

**Role** _administrator

### Validate the password of a user

**PUT /user/** (*username*) **/validate**
Validates the given password against the password of the user. Hashed passwords are assumed to be represented as hexadecimal strings. Another option to validate the user credentials is to call *whoami*.

Any hashed passwords need to be salted using the *salt of the user*.

**Query Parameters**

- **type** (*string*) –
  - `raw` - Password is in plaintext.

- `md5` (default) - Password is hashed.

**Status Codes**

- **`200 OK`** – The password is correct.
- **`403 Forbidden`** – The password is incorrect.

**Accepts**

- **`text/plain`** – The password of the user.

**Produces**

- **`text/plain`** – "OK <$PASSWORD>"

**Role** _administrator

### Retrieve the salt of a user

**`GET /user/`**(*username*)**`/password/salt`**
Retrieves the salt of the specified user.

**Status Codes**

- **`200`** – Salt is returned.
- **`204`** – No salt is set for the user.

**Produces**

- **`application/octet-stream`** – The salt of the user.

**Role** _administrator

### Generate a salt for a user

**`POST /user/`**(*username*)**`/password/salt`**
Generates a new salt for the user, overwriting any existing salt.

Note that this will invalidate any set password for the user and requires a new password to be set for the user to be able to login. This method is typically not relevant if passwords are updated using plaintext.

**Produces**

- **`application/octet-stream`** – The salt of the user.

**Role** _administrator

### Group-to-user relations

### List all groups for a user

**`GET /user/`**(*username*)**`/groups`**
Retrieves a list of all the groups a user belongs to.

**Query Parameters**

- **`allgroups`** (*`boolean`*) –
  - `true` - List all groups the user belongs to, including parent groups.
  - `false` (default) - Just list the groups that the user is directly assigned to.
- **`traverse`** (*`boolean`*) –

- `true` - When used in combination with `allgroups=true`, the groups' hierarchies are shown.
- `false` (default) - List the groups without hierarchical ordering.

**Produces**

- **`application/xml, application/json`** – GroupListDocument

**Role** _user_read

### List all roles for a user

**`GET /user/`** (*username*) **`/roles`**
Returns a list of all the roles a user has.

**Produces**

- **`application/xml, application/json`** – GroupListDocument

**Role** _user_read

### List all roles and groups for a user

**`GET /user/`** (*username*) **`/allgroups`**
Retrieves a list of all the groups a user belongs to, as well as all roles the user is in.

**Produces**

- **`application/xml, application/json`** – GroupListDocument

**Role** _user_read

### Add a user to multiple groups

**`PUT /user/`** (*username*) **`/groups`**
Adds a to multiple to groups. If the move parameter is set to `true`, it will cause the user to be moved to the specified groups.

**Query Parameters**

- **`move`** (*boolean*) –
  - `true` - Remove all previous group-to-user relations
  - `false` (default) - Keep the current group-to-user relations, and add the specified list

**Accepts**

- **`application/xml, application/json`** – GroupListDocument

**Role** _administrator

### Example

First the user belongs to a single group:

```
GET /user/myuser/groups
```

```
<GroupListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <group>
      <groupName>A</groupName>
      <role>false</role>
```

```
   </group>
</GroupListDocument>
```

The user is then added to groups B, C:

```
PUT /user/myuser/groups
Content-Type: application/xml

<GroupListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <group>
      <groupName>B</groupName>
   </group>
   <group>
      <groupName>C</groupName>
   </group>
</GroupListDocument>
```

```
GET /user/myuser/groups
```

```
<GroupListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <group>
      <groupName>A</groupName>
      <role>false</role>
   </group>
   <group>
      <groupName>B</groupName>
      <role>false</role>
   </group>
   <group>
      <groupName>C</groupName>
      <role>false</role>
   </group>
</GroupListDocument>
```

And then moved to groups A, B:

```
PUT /user/myuser/groups?move=true
Content-Type: application/xml

<GroupListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <group>
      <groupName>A</groupName>
   </group>
   <group>
      <groupName>B</groupName>
   </group>
</GroupListDocument>
```

```
GET /user/myuser/groups
```

```
<GroupListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <group>
      <groupName>A</groupName>
      <role>false</role>
   </group>
   <group>
```

```
      <groupName>B</groupName>
      <role>false</role>
   </group>
</GroupListDocument>
```

### User/group visualization

In order to easily see the dependencies between users and groups there is a functionality to render the user and group hierarchy as a graph. In order to render the graph, the Graphviz (http://www.graphviz.org/) package is required.

### Retrieve the user graph

**GET /user/graph**
Shows the relationships of users and groups.

**Query Parameters**

- **users** (*string*) – Comma-separated list of users to include. Default is all users.
- **groups** (*string*) – Comma-separated list of groups to include. Default is all groups.
- **disabled** (*string*) – If `true` only disabled users are shown, if `false` only enabled users are shown. Default is `all` - all users are shown.

**Produces**

- **image/png** –

**Role** _administrator

### Retrieve the user graph as dot file

**GET /user/graph/dot**
Shows the relationships of users and groups in dot format, for further processing.

**Query Parameters**

- **users** (*string*) – Comma-separated list of users to include. Default is all users.
- **groups** (*string*) – Comma-separated list of groups to include. Default is all groups.
- **disabled** (*string*) – If `true` only disabled users are shown, if `false` only enabled users are shown. Default is `all` - all users are shown.

**Produces**

- **text/plain, text/vnd.graphviz** –

**Role** _administrator

## 17.33.2 User access keys

User access keys are long-lived tokens that can be used to authenticate a user. Access keys do not expire, compared to *authentication tokens* that do.

A user can have multiple access keys, and can be used to allow multiple services to use the API without having to share the user password. Individual access keys can also be disabled to revoke access.

Note that the secret will only be shown when a key is first created.

### Managing access keys

### List all access keys for a user

**GET /user/** (*username*) **/key**
Retrieves the access keys for the specified user.

**Produces**

- **application/xml, application/json** – AccessKeyListDocument

**Role** _administrator

### Example

```
GET /user/stephen/key
Accept: application/xml

<AccessKeyListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <hits>1</hits>
  <key>
    <id>VSIDR62PJ4WSMJREGD6Z</id>
    <status>INACTIVE</status>
    <created>2018-06-01T10:31:17.275+02:00</created>
  </key>
</AccessKeyListDocument>
```

### Create an access key

**POST /user/** (*username*) **/key**
Generates a new access key and secret for the specified user. The only time the access key secret will be available is in the response of this request.

**Query Parameters**

- **name** (*string*) – A friendly name/description of the key.

**Produces**

- **application/xml, application/json** – AccessKeyDocument

**Role** _administrator

### Example

```
POST /user/stephen/key/
Accept: application/xml
```

```
<AccessKeyDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VSID2C56CQTA7DWW7DLF</id>
  <secret>PiFHewzGN3tZ/3lSeAkBVX+5YOw3wwpHTO2iqIaa</secret>
  <status>ACTIVE</status>
  <created>2018-06-01T10:36:13.891+02:00</created>
</AccessKeyDocument>
```

### Retrieve an access key

**GET /user/** (*username*) **/key/**
*key-id* Retrieves a specific access key.

**Produces**

- **application/xml, application/json** – AccessKeyDocument

**Role** _administrator

### Example

```
GET /user/stephen/key/VSID2C56CQTA7DWW7DLF
Accept: application/xml

<AccessKeyDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VSID2C56CQTA7DWW7DLF</id>
  <status>ACTIVE</status>
  <created>2018-06-01T10:36:13.891+02:00</created>
</AccessKeyDocument>
```

### Update an access key

**PUT /user/** (*username*) **/key/**
*key-id* Updates the name and/or status of a specific access key.

**Accepts**

- **application/xml, application/json** – AccessKeyDocument

**Produces**

- **application/xml, application/json** – AccessKeyDocument

**Role** _administrator

### Example

```
PUT /user/stephen/key/VSID2C56CQTA7DWW7DLF
Content-Type: application/xml

<AccessKeyDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <status>INACTIVE</status>
  <name>Public status dashboard</name>
</AccessKeyDocument>
```

```
<AccessKeyDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VSID2C56CQTA7DWW7DLF</id>
  <name>Public status dashboard</name>
  <status>INACTIVE</status>
  <created>2018-06-01T10:36:13.891+02:00</created>
</AccessKeyDocument>
```

### Delete an access key

**DELETE /user/** (*username*) **/key/**
*key-id* Removes the specific access key.

**Role** _administrator

#### Example

```
DELETE /user/stephen/key/VSID2C56CQTA7DWW7DLF
```

```
200 OK
```

### 17.33.3 User aliases

User aliases can be used to assign multiple usernames to a single user. These additional usernames are called aliases. Use aliases to for example allow a user to authenticate using either the users username or e-mail address.

Aliases are only ever looked up during authentication, and can not be used as a substitute for the usernames. In this way, user aliases are different to *external ids*.

#### Managing aliases

#### Create an alias

**PUT /user/** (*username*) **/alias/**
*name* Adds a new user alias for the specified user.

**Status Codes**

- **409 Conflict** – The alias is already used by another user.

**Role** _administrator

#### Example

```
PUT /user/stephen/alias/stephen@example.com
```

```
200 OK
```

```
GET API/user/stephen
```

```
<UserDocument xmlns="http://xml.vidispine.com/schema/vidispine" disabled="false">
  <userName>stephen</userName>
  <realName>Stephen</realName>
  <alias>stephen@example.com</alias>
</UserDocument>
```

#### Delete an alias

**DELETE /user/** (*username*) **/alias/**
*name* Removes the specific user alias.

**Role** _administrator

#### Example

```
DELETE /user/stephen/alias/stephen@example.com
```

```
200 OK
```

```
GET API/user/stephen
```

```
<UserDocument xmlns="http://xml.vidispine.com/schema/vidispine" disabled="false">
  <userName>stephen</userName>
  <realName>Stephen</realName>
</UserDocument>
```

### 17.33.4 User authentication tokens

User authentication tokens are short-lived tokens that can be used to authenticate a user. All tokens expire after a certain duration, but may auto-refresh on use to increase the expiration time of the token.

#### Token expiration

The rules for the expiration time depends on configuration property *userTokenMaxInterval* (default 60 seconds). If the expiration time is:

**Not specified** The token expires after the time entered in the configuration property *userTokenDefaultInterval* (default 60 seconds).

**Less than or equal to *userTokenMaxInterval*** Always allowed.

**Greater than *userTokenMaxInterval*** Only allowed if the *calling* user has `_administrator` role.

If `autoRefresh` is `true`, the expiration clock is reset with every API call when the token is used, with one exception. If the time since last reset is less than configuration property `userTokenRefreshInterval` (default 10 seconds), the token is not updated. This is in order to reduce database writes. Example:

1. Token is created, will expire in 60 seconds.
2. 8 seconds later, token is used. Since 8<10, token is not updated.
3. Another 8 seconds later, token is used again. Since 16>10, token is updated, and valid for 60 seconds more.

#### Managing tokens

#### Retrieve an authentication token

**GET /token**

Creates a authentication token for the calling user. This token can be used for calling the API without specifying username or password.

Useful when users authenticate using an *alias* and the actual username of the user is not known.

**Query Parameters**

- **seconds** (*integer*) – The duration of the token.
- **autoRefresh** (*boolean*) –
  - `true` - The expiration clock is reset with every API call.
  - `false` (default) - The token always expires after `seconds` seconds after the token was created.

**Status Codes**

- **409 Conflict** – The user is disabled.

**Produces**

- **application/xml, application/json** – AuthenticationTokenDocument: The generated token.
- **text/plain** – The generated token.

### Example

```
GET /token
Authorization: basic YWRtaW46YWRtaW4=
```

```
<AuthenticationTokenDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <token>5ay6Fxq2fFnmtVhrQq2owDvX0FE/RmdQG4SkefvW</token>
  <user>admin</user>
</AuthenticationTokenDocument>
```

## Retrieve an authentication token for a specific user

**GET /user/** (*username*) **/token**

Creates a authentication token for a user. This token can be used for calling the API without specifying username or password.

The username path parameter must match the calling user's credentials, unless the calling user has `_administrator` role.

**Query Parameters**

- **seconds** (*integer*) – The duration of the token.
- **autoRefresh** (*boolean*) –
  - `true` - The expiration clock is reset with every API call.
  - `false` (default) - The token always expires after `seconds` seconds after the token was created.

**Status Codes**

- **409 Conflict** – The user is disabled.

**Produces**

- **text/plain** – The generated token.

### Example

```
GET /user/myuser/token
Authorization: basic YWRtaW46YWRtaW4=
```

```
6663e105-828e-45c1-ac54-7dd17f3e8a38
```

```
GET /item
Authorization: token 6663e105-828e-45c1-ac54-7dd17f3e8a38
```

This will return items that user `myuser` has access to.

# 17.34 Vidinet

## 17.34.1 Cost estimation

Retrieve cost estimates from Vidinet.

The following resources support cost estimation:

- *POST /item/(item-id)/transcode*
- *POST /item/(item-id)/shape/(shape-id)/analyze*
- *POST /import*
- *POST /storage/(storage-id)/file/(file-id)/import*
- *POST /storage/file/(file-id)/import*

### Request a cost estimate

**POST /cost/** (*path*)

Requests a cost estimate for performing a specific operation.

The same query parameters and request body as used by the target request should be specified in the cost estimate request.

No additional roles or permissions are required to request a cost estimate. If a user has permission to for example transcode an item, then that user may also request a cost estimate for that operation.

**Produces**

- **application/xml, application/json** – CostEstimateDocument

### Retrieve the cost estimate results

**GET /cost/estimate/** (*id*)

Retrieves the cost estimate.

**Produces**

- **application/xml, application/json** – CostEstimateDocument

### Examples

#### Item transcode cost estimation

Requesting a cost estimate for transcoding an item to __mp4 using the Vidinet TRANSCODE resource VX-5:

```
POST /cost/item/VX-45/transcode?tag=__mp4&resource=VX-5
```

```
<CostEstimateDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>dGVzdA==</id>
  <url>http://localhost:8080/API/cost/estimate/dGVzdA==</url>
</CostEstimateDocument>
```

Retrieving the cost estimate status and amount.

```
GET /cost/estimate/dGVzdA==
```

```
<CostEstimateDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>dGVzdA==</id>
  <url>http://localhost:8080/API/cost/estimate/dGVzdA==</url>
  <state>FINISHED</state>
  <service>
    <resource>VX-5</resource>
    <type>TRANSCODER</type>
    <state>ONLINE</state>
  </service>
</CostEstimateDocument>
```

## 17.35 Vidispine logs

Extract logs from the system.

### 17.35.1 Retrieving log files

If your application has a custom form for reporting issues then you can collect the log files using the `vidispine-logs` resource.

#### Retrieve log files

**`GET /vidispine-logs`**
Retrieves a zip file with the various service log files.

**Query Parameters**

- **`starttime`** (*`string`*) – Required. ISO 8601 start time of time span.
- **`endtime`** (*`string`*) – Required. ISO 8601 end time of time span.
- **`comment`** (*`string`*) – Detailed description of what the problem is, to be written in zip file.
- **`user`** (*`string`*) – Comma-separated list of user names to retrieve information about.
- **`storage`** (*`string`*) – Comma-separated list of storage ids to retrieve information about.
- **`item`** (*`string`*) – Comma-separated list of item ids to retrieve information about.
- **`job`** (*`string`*) – Comma-separated list of job ids to retrieve information about.
- **`selftest`** (*`boolean`*) – If `true`, includes the result of a self-test. Default is `false`.

**Produces**

- **`application/zip`** – A zip file with various log files collated.
- **`multipart/form-data , multipart/mixed`** – A multi-part response, with parts:
  - `zip` Containing the zip file.
  - `message-text` Containing CRLF-separated list of warnings.
  - `message-json` Containing JSON array list of warnings.

**Role** _administrator

### 17.35.2 Log retrieval jobs

Instead of a synchronous call, Vidispine can also start a *pseudo-job* that creates the log report in the background. This is not a real Vidispine job, so there are no priorities, notifications, etc.

In case of a restart of Vidispine, the job is aborted. Finished jobs are stored in a temporary location, however.

### Start a log report job

**POST /vidispine-logs/job**
Starts a log report job that collects service log files into a zip file.

**Query Parameters**

- **starttime** (*string*) – Required. ISO 8601 start time of time span.
- **endtime** (*string*) – Required. ISO 8601 end time of time span.
- **comment** (*string*) – Detailed description of what the problem is, to be written in zip file.
- **user** (*string*) – Comma-separated list of user names to retrieve information about.
- **storage** (*string*) – Comma-separated list of storage ids to retrieve information about.
- **item** (*string*) – Comma-separated list of item ids to retrieve information about.
- **job** (*string*) – Comma-separated list of job ids to retrieve information about.
- **selftest** (*boolean*) – If `true`, includes the result of a self-test. Default is `false`.

**Produces**

- **application/xml, application/json** – A JobDocument that describes the job.

**Role** _administrator

### List all log report jobs

**GET /vidispine-logs/job**
Return information about all jobs that have not expired.

**Produces**

- **application/xml, application/json** – JobListDocument

**Role** _administrator

### Retrieve a log report job

**GET /vidispine-logs/job/** (*job-id*)
Return information about specified job.

**Produces**

- **application/xml, application/json** – JobDocument

**Role** _administrator

### Retrieve job result

**GET /APInoauth/vidispine-logs/job/** (*job-id*) **.zip**
Returns the zip file generated by a finished job.

**Role** _administrator

### Delete a log report job

**DELETE /vidispine-logs/job/** (*job-id*)
Deletes a job and the report created.

**Role** _administrator

### 17.35.3 Upload of logs to Vidispine

If a log has been created by a job, Vidispine can upload it to vidispine.com. In addition, source files referenced by items entered in the log report can be attached. This is a two-phase operation, first query the API of what files there are, then start the job.

Status of upload is included in the log report job, see above.

#### Retrieve files in job

**GET /vidispine-logs/job/** (*job-id*) **/uri**
Return file information about specified job.

**Produces**

- **application/json** – An array of objects with keys name, `displayUri`, `uri`, and `size`.

**Role** _administrator

#### Start an upload

**POST /vidispine-logs/job/** (*job-id*) **/upload**
Starts upload.

**Accepts**

- **application/xml, application/json** – A URIListDocument with URIs from the list of files in the log.

**Role** _administrator

#### Abort an upload

**DELETE /vidispine-logs/job/** (*job-id*) **/upload**
Aborts upload process.

**Role** _administrator

## 17.36 Vidispine services

Periodic and background operations are executed internally by Vidispine services. These resources are meant for system administrators or developers that need to troubleshoot or monitor a server.

The available Vidispine services may vary from version to version and should not be considered to be part of the stable API.

**Note:** This is an internal API, and may change in future versions.

### 17.36.1 Vidispine services

#### List all services

**GET /vidispine-service**
Return all Vidispine services and their status and load.

**Query Parameters**

- **stacktrace** (*boolean*) – Return a stack trace for each service. Default is `false`

**Produces**

- **application/xml, application/json** – VidispineServiceListDocument

**Role** _administrator

#### Retrieve a service

**GET /vidispine-service/service/** (*service*)
Return a specific Vidispine service and its status and load.

**Query Parameters**

- **stacktrace** (*boolean*) – Return a stack trace for each service. Default is `false`

**Produces**

- **application/xml, application/json** – VidispineServiceListDocument

**Role** _administrator

### 17.36.2 Service status

#### Enable all services

**PUT /vidispine-service/enable**
Enables all disabled services.

**Role** _administrator

#### Enable a service

**PUT /vidispine-service/service/** (*service*) **/enable**
Enables this service if disabled.

**Role** _administrator

#### Disable all services

**PUT /vidispine-service/disable**
Disables all services. No instance in the cluster will run any services.

**Role** _administrator

#### Disable a service

**PUT /vidispine-service/service/** (*service*) **/disable**
Disables this service. No instance in the cluster will run this service.

**Role** _administrator

### 17.36.3 Stack trace

#### Retrieve service stack trace

**GET /vidispine-service/stacktrace**
Returns a stack trace from the system.

**Produces**

- **text/plain** – A stack trace in plain text.

**Role** _administrator

## 17.37 Vidispine server agents

Vidispine server agents (VSAs) are typically handled with the `vidispine-admin` and `vidispine-agent-admin` shell tools. A few API commands are available for control, though.

### 17.37.1 Managing VSAs

#### List all server agents

**GET /vxa**
Returns all VSAs.

**Produces**

- **application/xml, application/json** – A VXAListDocument

**Role** _vxa_read

#### Retrieve a server agent

**GET /vxa/** (*uuid*)
Retrieves a specific VSA. As an alternative to the UUID, the name of the VSA can be used instead. The name syntax only works if the name is unique among the VSAs.

**Produces**

- **application/xml, application/json** – A VXADocument

**Role** _vxa_read

#### Retrieve the server agent configuration

**GET /vxa/** (*uuid*) **/configuration**
Returns the client-side configuration of the VSA.

**Produces**

- **application/json** – A JSON object with the settings and current status of the VSA.

**Role** _vxa_read

#### Register a server agent

**PUT /vxa/enable-ssh**
Registers a new VSA node in the system.

**Query Parameters**

- **vsip** (*string[]*) – The address to which the VSA should connect. Can be specified multiple times for a cluster configuration.
- **vsport** (*string[]*) – The port to which the VSA should connect. Can be specified multiple times for a cluster configuration.
- **ws** (*string[]*) – The URI to the API endpoint. Used to enable WebSocket tunneling, as an alternative to `vsip`/`vsport`. Can be specified multiple times for a cluster configuration.
- **uuid** (*string*) – The UUID of the VSA. If not set, Vidispine will assign a UUID.
- **vxaname** (*string*) – The name of the VSA.

**Produces**

- **text/plain** – A text configuration to be added on the VSA's configuration.

**Role** _administrator

### Delete a server agent

**DELETE /vxa/** (*uuid*)
Removes the VSA from the system

**Role** _administrator

## 17.38 XML Schema

This is the XML schema used to define data types in the Vidispine API. For a snapshot of the XML schema, see http://xml.vidispine.com/schema/vidispine/

### 17.38.1 xmlSchema.xsd

API specific schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://xml.vidispine.com/schema/vidispine"
           elementFormDefault="qualified"
           xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
           jaxb:version="1.0"
           xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
           jaxb:extensionBindingPrefixes="xjc" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine">

    <xs:include schemaLocation="common.xsd"/>
    <xs:include schemaLocation="transcoder.xsd"/>

    <xs:annotation>
        <xs:appinfo>
            <jaxb:globalBindings generateIsSetMethod="true">
                <xjc:serializable uid="10000"/>
                <!--<xjc:typeSubstitution type="complex"/>-->
            </jaxb:globalBindings>
        </xs:appinfo>
    </xs:annotation>

    <xs:complexType name="AmountType">
        <xs:simpleContent>
            <xs:extension base="xs:decimal">
                <xs:attribute name="unit" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="AnalyzeAudioJobType">
        <xs:sequence>
            <xs:element name="otif" type="tns:OtifPresetType" minOccurs="0" maxOccurs=
↪"unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AnalyzeVideoJobType">
        <xs:sequence>
            <xs:element name="otif" type="tns:OtifPresetType" minOccurs="0" maxOccurs=
↪"unbounded"/>
```

```
        </xs:sequence>
    </xs:complexType>
```

*AnalyzeJobDocument*

```
    <xs:element name="AnalyzeJobDocument" type="tns:AnalyzeJobType" xmlns:tns="http://xml.vidispine.com/schema/vidispine"/>
    <xs:complexType name="AnalyzeJobType">
        <xs:sequence>
            <xs:element name="black" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="threshold" type="xs:float" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="percentage" type="xs:int" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="bars" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="threshold" type="xs:float" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="percentage" type="xs:int" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="freeze" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="threshold" type="xs:float" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="time" type="xs:float" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <xs:element name="channel" type="tns:AnalyzeAudioChannelType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="audio" type="tns:AnalyzeAudioJobType" minOccurs="0" maxOccurs="1"/>
        <xs:element name="video" type="tns:AnalyzeVideoJobType" minOccurs="0" maxOccurs="1"/>
        <xs:element name="highlighter" type="tns:HighlighterType" minOccurs="0" maxOccurs="1"/>
        <xs:element name="smartcrop" type="tns:SmartCropType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SearchResultEntryTimespanType">
        <xs:sequence>
            <xs:element name="field" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
```

```
                        <xs:element name="name" type="xs:string" minOccurs="1"␣
↪maxOccurs="1"/>
                        <xs:element name="value" type="xs:string" minOccurs="0"␣
↪maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="start" type="xs:string" use="required"/>
        <xs:attribute name="end" type="xs:string" use="required"/>
    </xs:complexType>

    <xs:complexType name="SearchResultEntryType">
        <xs:sequence>
            <xs:choice>
                <xs:element name="item" type="tns:ItemType"/>
                <xs:element name="collection" type="tns:CollectionType"/>
                <xs:element name="shape" type="tns:ShapeType"/>
                <xs:element name="file" type="tns:FileType"/>
            </xs:choice>
            <xs:element name="timespan" type="tns:SearchResultEntryTimespanType"␣
↪minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="start" type="xs:string" use="optional"/>
        <xs:attribute name="end" type="xs:string" use="optional"/>
        <xs:attribute name="type" type="xs:string" use="optional"/>
        <xs:attribute name="id" type="xs:string" use="optional"/>
        <xs:attribute name="parent_type" type="xs:string" use="optional"/>
        <xs:attribute name="parent_id" type="xs:string" use="optional"/>
        <xs:attribute name="base" type="xs:string" use="optional"/>
    </xs:complexType>
```

*SearchResultDocument*

```
    <xs:element name="SearchResultDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:SearchResultType"/>
    <xs:complexType name="SearchResultType">
        <xs:sequence>
            <xs:element name="hits" minOccurs="0" maxOccurs="1" type="xs:int"/>
            <xs:element name="suggestion" minOccurs="0" maxOccurs="unbounded" type=
↪"tns:SuggestionResultType"/>
            <xs:element name="autocomplete" minOccurs="0" maxOccurs="unbounded" type=
↪"tns:AutocompleteResponseType"/>
            <xs:element name="entry" type="tns:SearchResultEntryType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="facet" minOccurs="0" maxOccurs="unbounded" type="tns:
↪FacetType"/>
            <xs:element name="nextCursor" type="xs:string" maxOccurs="1" minOccurs="0
↪"></xs:element>
        </xs:sequence>
    </xs:complexType>
```

*MetadataEntryListDocument*

```
    <xs:element name="MetadataEntryListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:MetadataEntryListType2"/>
    <xs:complexType name="MetadataEntryListType2">
        <xs:sequence>
```

```
            <xs:element name="entry" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="tns:MetadataEntryType">
                            <xs:attribute name="uuid" type="xs:string"/>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*MetadataEntryDocument*

```
    <xs:element name="MetadataEntryDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:MetadataEntryType" />
    <xs:complexType name="MetadataEntryType">
        <xs:sequence>
            <xs:element name="group" type="tns:MetadataGroupValueType" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="field" type="tns:MetadataFieldValueType" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="value" type="tns:MetadataValueType" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="source" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="xs:string" minOccurs="1"␣
↪maxOccurs="1"/>
                        <xs:element name="type" type="xs:string" minOccurs="1"␣
↪maxOccurs="1"/>
                        <xs:element name="loc" type="xs:anyURI" minOccurs="1"␣
↪maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*MetadataSchemaDocument*

```
    <xs:element name="MetadataSchemaDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:MetadataSchemaType" />
    <xs:complexType name="MetadataSchemaType">
        <xs:sequence>
           <xs:element name="group" minOccurs="0" maxOccurs="unbounded" type="tns:
↪MetadataSchemaGroupType"/>
        </xs:sequence>
    </xs:complexType>
```

*MetadataSchemaGroupDocument*

```
    <xs:element name="MetadataSchemaGroupDocument" xmlns:tns="http://xml.vidispine.
↪com/schema/vidispine" type="tns:MetadataSchemaGroupType" />
    <xs:complexType name="MetadataSchemaGroupType">
       <xs:sequence>
            <xs:element name="group" minOccurs="0" maxOccurs="unbounded" type="tns:
↪MetadataSchemaElementType"/>
```

```
            <xs:element name="field" minOccurs="0" maxOccurs="unbounded" type="tns:
↪MetadataSchemaElementType"/>
        </xs:sequence>
        <xs:attributeGroup ref="tns:MetadataSchemaAttributes"/>
    </xs:complexType>
```

*BeanCallbackListDocument*

```
    <xs:element name="BeanCallbackListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:BeanCallbackListType"/>
    <xs:complexType name="BeanCallbackListType">
        <xs:sequence>
            <xs:element name="callback" type="tns:BeanCallbackType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*BeanCallbackDocument*

```
    <xs:element name="BeanCallbackDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:BeanCallbackType"/>
    <xs:complexType name="BeanCallbackType">
        <xs:sequence>
            <xs:element name="sourceBean" type="xs:string" minOccurs="1" maxOccurs="1
↪"/>
            <xs:element name="sourceMethod" type="xs:string" minOccurs="1" maxOccurs=
↪"1"/>
            <xs:element name="destinationBean" type="xs:string" minOccurs="1"␣
↪maxOccurs="1"/>
            <xs:element name="destinationMethod" type="xs:string" minOccurs="1"␣
↪maxOccurs="1"/>
            <xs:element name="lastSuccess" type="xs:dateTime" minOccurs="0" maxOccurs=
↪"1"/>
            <xs:element name="lastFailure" type="xs:dateTime" minOccurs="0" maxOccurs=
↪"1"/>
            <xs:element name="errorMessage" type="xs:string" minOccurs="0" maxOccurs=
↪"1"/>
        </xs:sequence>
    </xs:complexType>
```

*AuditLogDocument*

```
    <xs:element name="AuditLogDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:AuditLogType"/>
    <xs:complexType name="AuditLogType">
        <xs:sequence>
            <xs:element name="count" type="xs:long" minOccurs="0" maxOccurs="1"/>
            <xs:element name="entry" type="tns:AuditLogEntryType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AuditLogEntryType">
        <xs:sequence>
            <xs:element name="username" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="method" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="path" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="queryParameters" type="xs:string" minOccurs="1"␣
↪maxOccurs="1"/>
```

```
            <xs:element name="matrixParameters" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="runAs" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="contentType" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="contentLength" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="body" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="responseCode" type="xs:int" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="timestamp" type="xs:dateTime" use="required"/>
    </xs:complexType>
```

*ConfigurationPropertyListDocument*

```
    <xs:element name="ConfigurationPropertyListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ConfigurationPropertyListType"/>
    <xs:complexType name="ConfigurationPropertyListType">
        <xs:sequence>
            <xs:element name="property" type="tns:ConfigurationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*ConfigurationPropertyDocument*

```
    <xs:element name="ConfigurationPropertyDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ConfigurationPropertyType"/>
    <xs:complexType name="ConfigurationPropertyType">
        <xs:sequence>
            <xs:element name="key" minOccurs="1" maxOccurs="1" type="xs:string"/>
            <xs:element name="value" minOccurs="0" maxOccurs="1" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="lastChange" type="xs:dateTime" use="optional"/>
    </xs:complexType>

    <xs:complexType name="CollectionReorderEntryType">
        <xs:attribute name="id" use="optional" type="xs:string"/>
        <xs:attribute name="reference" use="optional" type="xs:string"/>
        <xs:attribute name="before" use="optional" type="xs:string"/>
        <xs:attribute name="after" use="optional" type="xs:string"/>
    </xs:complexType>
```

*CollectionReorderDocument*

```
    <xs:element name="CollectionReorderDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:CollectionReorderType"/>
    <xs:complexType name="CollectionReorderType">
        <xs:sequence>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="item" type="tns:CollectionReorderEntryType"/>
                <xs:element name="collection" type="tns:CollectionReorderEntryType"/>
                <xs:element name="library" type="tns:CollectionReorderEntryType"/>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
```

*ExternalIdentifierNamespaceListDocument*

```
    <xs:element name="ExternalIdentifierNamespaceListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ExternalIdentifierNamespaceListType"/>
    <xs:complexType name="ExternalIdentifierNamespaceListType">
        <xs:sequence>
            <xs:element name="namespace" type="tns:ExternalIdentifierNamespaceType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*ExternalIdentifierNamespaceDocument*

```
    <xs:element name="ExternalIdentifierNamespaceDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ExternalIdentifierNamespaceType"/>
    <xs:complexType name="ExternalIdentifierNamespaceType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="pattern" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="priority" type="xs:int" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*ExternalIdentifierListDocument*

```
    <xs:element name="ExternalIdentifierListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ExternalIdentifierListType"/>
    <xs:complexType name="ExternalIdentifierListType">
        <xs:sequence>
            <xs:element name="id" type="tns:ExternalIdentifierType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*ExternalIdentifierDocument*

```
    <xs:element name="ExternalIdentifierDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ExternalIdentifierType"/>
    <xs:complexType name="ExternalIdentifierType">
        <xs:sequence>
            <xs:element name="entityId" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="entityType" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="namespace" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="externalId" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*MetadataFieldResultDocument*

```
    <xs:element name="MetadataFieldResultDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:MetadataFieldResultType"/>

    <xs:complexType name="MetadataFieldResultType">
        <xs:sequence>
            <xs:element name="hits" minOccurs="1" maxOccurs="1" type="xs:int"/>
            <xs:element name="group" minOccurs="0" maxOccurs="unbounded">
```

```
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="value" type="tns:MetadataGroupValueType" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="definition" type="tns:MetadataFieldGroupType" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="source" minOccurs="0" maxOccurs="1">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="id" type="xs:string" minOccurs="1" maxOccurs="1"/>
                                    <xs:element name="type" type="xs:string" minOccurs="1" maxOccurs="1"/>
                                    <xs:element name="loc" type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                    <xs:attribute name="name" type="xs:string" use="required"/>
                    <xs:attribute name="uuid" type="xs:string" use="required"/>
                    <xs:attribute name="start" type="xs:string" use="required"/>
                    <xs:attribute name="end" type="xs:string" use="required"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*MetadataFieldGroupListDocument*

```
    <xs:element name="MetadataFieldGroupListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:MetadataFieldGroupListType"/>
    <xs:complexType name="MetadataFieldGroupListType">
        <xs:sequence>
            <xs:element name="group" type="tns:MetadataFieldGroupType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*MetadataFieldGroupDocument*

```
    <xs:element name="MetadataFieldGroupDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:MetadataFieldGroupType"/>
    <xs:complexType name="MetadataFieldGroupType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="schema" type="tns:MetadataSchemaElementType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="data" minOccurs="0" maxOccurs="unbounded" type="tns:KeyValuePairType"/>
            <xs:element name="field" type="tns:MetadataFieldType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="group" type="tns:MetadataFieldGroupType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="access" minOccurs="0" maxOccurs="unbounded" type="tns:MetadataFieldAccessControlType"/>
            <xs:element name="externalId" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
```

```
            <xs:element name="origin" type="xs:string" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
        <xs:attribute name="inheritance" type="xs:string" use="optional"/>
    </xs:complexType>
```

*MetadataFieldListDocument*

```
    <xs:element name="MetadataFieldListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:MetadataFieldListType"/>
    <xs:complexType name="MetadataFieldListType">
        <xs:sequence>
            <xs:element name="access" minOccurs="0" maxOccurs="unbounded" type="tns:MetadataFieldAccessControlType"/>
            <xs:element name="field" minOccurs="0" maxOccurs="unbounded" type="tns:MetadataFieldType"/>
        </xs:sequence>
    </xs:complexType>
```

*MetadataFieldAccessControlListDocument*

```
    <xs:element name="MetadataFieldAccessControlListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:MetadataFieldAccessControlListType"/>
    <xs:complexType name="MetadataFieldAccessControlListType">
        <xs:sequence>
            <xs:element name="access" type="tns:MetadataFieldAccessControlType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*MetadataFieldAccessControlDocument*

```
    <xs:element name="MetadataFieldAccessControlDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:MetadataFieldAccessControlType"/>
    <xs:complexType name="MetadataFieldAccessControlType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:choice minOccurs="0" maxOccurs="1">
                <xs:element name="field" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="fieldGroup" type="xs:string" minOccurs="1" maxOccurs="1"/>
            </xs:choice>
            <xs:element name="user" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="group" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="permission" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*add*

```
    <xs:element name="add" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SolrAddType"/> <!-- notoc -->
    <xs:complexType name="SolrAddType"> <!-- notoc -->
        <xs:sequence>
            <xs:element name="doc" type="tns:SolrDocumentType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
```

```
    </xs:complexType>
```

*doc*

```
    <xs:element name="doc" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SolrDocumentType"/> <!-- notoc -->
    <xs:complexType name="SolrDocumentType"> <!-- notoc -->
        <xs:sequence>
            <xs:element name="field" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attribute name="name" type="xs:string" use="required"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*LockDocument*

```
    <xs:element name="LockDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:LockType"/>
    <xs:complexType name="LockType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="user" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="expires" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*ExceptionDocument*

```
    <xs:element name="ExceptionDocument" type="tns:ExceptionType" />
    <xs:complexType name="ExceptionType">
        <xs:choice>
            <xs:element name="notFound" type="tns:NotFoundExceptionType"/>
            <xs:element name="internalServer" type="tns:InternalServerExceptionType"/>
            <xs:element name="forbidden" type="tns:ForbiddenExceptionType"/>
            <xs:element name="notYetImplemented" type="tns:NotYetImplementedExceptionType"/>
            <xs:element name="conflict" type="tns:ConflictExceptionType"/>
            <xs:element name="invalidInput" type="tns:InvalidInputExceptionType"/>
            <xs:element name="licenseFault" type="tns:LicenseExceptionType"/>
            <xs:element name="fileAlreadyExists" type="tns:FileAlreadyExistsExceptionType"/>
            <xs:element name="notAuthorized" type="tns:NotAuthorizedExceptionType"/>
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="NotFoundExceptionType">
        <xs:sequence>
            <xs:element name="type" minOccurs="0" type="xs:string" />
            <xs:element name="id" minOccurs="0" type="xs:string" />
            <xs:element name="context" minOccurs="0" type="xs:string" />
```

```
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="LicenseExceptionType">
        <xs:sequence>
            <xs:element name="message" minOccurs="0" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="InternalServerExceptionType">
        <xs:sequence>
            <xs:element name="message" minOccurs="0" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ForbiddenExceptionType">
        <xs:sequence>
            <xs:element name="context" minOccurs="0" type="xs:string" />
            <xs:element name="id" minOccurs="0" type="xs:string" />
            <xs:element name="explanation" minOccurs="0" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="NotYetImplementedExceptionType">
        <xs:sequence>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ConflictExceptionType">
        <xs:sequence>
            <xs:element name="context" minOccurs="0" type="xs:string" />
            <xs:element name="id" minOccurs="0" type="xs:string" />
            <xs:element name="explanation" minOccurs="0" type="xs:string" />
            <xs:element name="value" minOccurs="0" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="InvalidInputExceptionType">
        <xs:sequence>
            <xs:element name="context" minOccurs="0" type="xs:string" />
            <xs:element name="id" minOccurs="0" type="xs:string" />
            <xs:element name="explanation" minOccurs="0" type="xs:string" />
            <xs:element name="value" minOccurs="0" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FileAlreadyExistsExceptionType">
        <xs:sequence>
            <xs:element name="storageId" minOccurs="0" type="xs:string" />
            <xs:element name="fileId" minOccurs="0" type="xs:string" />
            <xs:element name="path" minOccurs="0" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="NotAuthorizedExceptionType">
        <xs:sequence>
            <xs:element name="message" minOccurs="0" type="xs:string" />
        </xs:sequence>
```

```
    </xs:complexType>
```

*AccessControlMergedGroupDocument*

```
    <xs:element name="AccessControlMergedGroupDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:AccessControlMergedGroupType"/>
    <xs:complexType name="AccessControlMergedGroupType">
        <xs:sequence>
            <xs:element name="access" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="group" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="grantor" type="xs:string" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="permission" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="type" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="extradata" type="xs:string" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="collection" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="library" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="originalDisabledGrantor" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="rank" use="optional" type="xs:int"/>
                    <xs:attribute name="id" use="optional" type="tns:SiteIdType"/>
                    <xs:attribute name="effectivePermission" use="optional" type="xs:string"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*AccessControlMergedDocument*

```
    <xs:element name="AccessControlMergedDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:AccessControlMergedType"/>
    <xs:complexType name="AccessControlMergedType">
        <xs:sequence>
            <xs:element name="query" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="username" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="permission" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="type" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="extradata" type="xs:string" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="item" type="tns:SiteIdType" minOccurs="1" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
```

```
            </xs:element>
            <xs:element name="access" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="grantor" type="xs:string" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="permission" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="type" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="extradata" type="xs:string" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="group" type="xs:string" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="collection" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="library" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="originalDisabledGrantor" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="superUser" use="optional" type="xs:boolean"/>
                    <xs:attribute name="rank" use="required" type="xs:int"/>
                    <xs:attribute name="matches" use="optional" type="xs:boolean"/>
                    <xs:attribute name="id" use="optional" type="tns:SiteIdType"/>
                    <xs:attribute name="username" use="optional" type="xs:string"/>
                    <xs:attribute name="effectivePermission" use="optional" type="xs:string"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="fieldGroup" type="tns:MetadataFieldGroupPermissionType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="field" type="tns:MetadataFieldPermissionType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="MetadataFieldGroupPermissionType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="permission" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="fieldGroup" type="tns:MetadataFieldGroupPermissionType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="field" type="tns:MetadataFieldPermissionType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="username" use="required" type="xs:string"/>
    </xs:complexType>

    <xs:complexType name="MetadataFieldPermissionType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="permission" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="username" use="required" type="xs:string"/>
    </xs:complexType>
```

*ImportSettingsDocument*

```
    <xs:element name="ImportSettingsDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ImportSettingsType"/>

    <xs:complexType name="ImportSettingsType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="access" type="tns:AccessControlType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*ScheduledRequestDocument*

```
    <xs:element name="ScheduledRequestDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ScheduledRequestType"/>
```

*ScheduledRequestListDocument*

```
    <xs:element name="ScheduledRequestListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ScheduledRequestListType"/>

    <xs:complexType name="ScheduledRequestListType">
        <xs:sequence>
            <xs:element name="scheduledRequest" type="tns:ScheduledRequestType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ScheduledRequestType">
        <xs:sequence>
            <xs:element name="id" type="xs:long" minOccurs="1" maxOccurs="1"/>
            <xs:element name="user" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="state" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="date" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
            <xs:element name="created" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
            <xs:element name="executed" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="request" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="uri" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="method" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="body" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="response" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="statusCode" type="xs:int" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="hasBody" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
```

```
                        <xs:element name="contentType" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*LibrarySettingsDocument*

```
    <xs:element name="LibrarySettingsDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:LibrarySettingsType"/>
    <xs:complexType name="LibrarySettingsType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="username" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="updateMode" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="autoRefresh" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="updateFrequency" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="lastUpdate" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="query" type="tns:ItemSearchType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*ImportAccessControlListDocument*

```
    <xs:element name="ImportAccessControlListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ImportAccessControlListType"/>
    <xs:complexType name="ImportAccessControlListType">
        <xs:sequence>
            <xs:element name="group" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="permission" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*StorageGroupListDocument*

```
    <xs:element name="StorageGroupListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:StorageGroupListType"/>
    <xs:complexType name="StorageGroupListType">
        <xs:sequence>
            <xs:element name="group" type="tns:StorageGroupType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*StorageGroupDocument*

```
    <xs:element name="StorageGroupDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:StorageGroupType" />
    <xs:complexType name="StorageGroupType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="data" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="key" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="value" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="storage" type="tns:StorageType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*ProjectDocument*

```
    <xs:element name="ProjectDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ProjectType"/>
    <xs:complexType name="ProjectType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="metadata" type="tns:MetadataType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*ProjectVersionDocument*

```
    <xs:element name="ProjectVersionDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ProjectVersionType"/>
    <xs:complexType name="ProjectVersionType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="externalId" type="xs:string" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="uri" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="sequence" type="tns:SequenceType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="metadata" type="tns:MetadataType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

```
678
679     <xs:complexType name="ExternalMediaType">
680         <xs:sequence>
681             <xs:element name="uri" type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
682             <xs:element name="format" type="xs:string" minOccurs="1" maxOccurs="1"/>
683             <xs:element name="essenceStreamId" default="0" type="xs:int" maxOccurs="1
↪"/>
684             <xs:element name="timeBase" type="tns:TimeBaseType" minOccurs="1"␣
↪maxOccurs="1"/>
685             <xs:element name="samples" type="xs:integer" minOccurs="1" maxOccurs="1"/>
686         </xs:sequence>
687     </xs:complexType>
688
689     <xs:complexType name="ExternalVideoMediaType">
690         <xs:complexContent>
691             <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
↪"tns:ExternalMediaType">
692                 <xs:sequence>
693                     <xs:element name="width" minOccurs="1" maxOccurs="1" type="xs:int
↪"/>
694                     <xs:element name="height" minOccurs="1" maxOccurs="1" type="xs:int
↪"/>
695                     <xs:element name="pixelAspectRatio" type="tns:AspectRatioType"␣
↪minOccurs="0" maxOccurs="1"/>
696                 </xs:sequence>
697             </xs:extension>
698         </xs:complexContent>
699     </xs:complexType>
700
701     <xs:complexType name="SequenceMediaType">
702         <xs:sequence>
703             <xs:choice minOccurs="1" maxOccurs="1">
704                 <xs:element name="externalVideoMedia" type="tns:ExternalVideoMediaType
↪" minOccurs="1" maxOccurs="1"/>
705                 <xs:element name="item" type="tns:SiteIdType" minOccurs="1" maxOccurs=
↪"1"/>
706             </xs:choice>
707             <xs:element name="sourceTrack" minOccurs="1" maxOccurs="1" type="xs:int"/>
708             <xs:element name="in" minOccurs="0" maxOccurs="1" type="tns:TimeCodeType"/
↪>
709             <xs:element name="out" minOccurs="0" maxOccurs="1" type="tns:TimeCodeType
↪"/>
710             <xs:element name="sourceIn" minOccurs="0" maxOccurs="1" type="tns:
↪TimeCodeType"/>
711             <xs:element name="sourceOut" minOccurs="0" maxOccurs="1" type="tns:
↪TimeCodeType"/>
712             <xs:element name="effect" type="tns:EffectType" minOccurs="0" maxOccurs=
↪"unbounded"/>
713             <xs:element name="reference" type="xs:string" minOccurs="0" maxOccurs="1"/
↪>
714         </xs:sequence>
715     </xs:complexType>
716
717     <!-- Like TransitionType, except that it uses in and out points and has user␣
↪friendly color values -->
718     <xs:complexType name="SequenceTransitionType">
719         <xs:sequence>
720             <xs:element name="in" type="tns:TimeCodeType" minOccurs="1" maxOccurs="1"/
↪>
```

```
            <xs:element name="out" type="tns:TimeCodeType" minOccurs="1" maxOccurs="1
↪"/>
            <xs:element name="wipe" type="xs:int" minOccurs="0"/>
            <xs:element name="transition" type="xs:string"/>
            <xs:element name="horizRepeat" type="xs:int" minOccurs="0"/>
            <xs:element name="vertRepeat" type="xs:int" minOccurs="0"/>
            <xs:element name="startPercentage" type="xs:int" minOccurs="0"/>
            <xs:element name="endPercentage" type="xs:int" minOccurs="0"/>
            <xs:element name="reverse" type="xs:boolean" minOccurs="0"/>
            <xs:element name="borderWidth" type="xs:int" minOccurs="0"/>
            <xs:element name="borderColor" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
```

*SequenceListDocument*

```
    <xs:element name="SequenceListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:SequenceListType" />
    <xs:complexType name="SequenceListType">
        <xs:sequence>
            <xs:element name="sequence" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="tns:SiteIdType" minOccurs="1"␣
↪maxOccurs="1"/>
                        <xs:element name="type" type="xs:string" minOccurs="1"␣
↪maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*SequenceDocument*

```
    <xs:element name="SequenceDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:SequenceType" />
    <xs:complexType name="SequenceType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="track" type="tns:SequenceTrackType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="override" type="tns:TranscodePresetType" minOccurs="0"␣
↪maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SequenceTrackType">
        <xs:sequence>
            <xs:element name="audio" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="segment" type="tns:SequenceMediaType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="transition" type="tns:SequenceTransitionType" minOccurs=
↪"0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*JobProblemListDocument*

```
    <xs:element name="JobProblemListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:JobProblemListType" />
    <xs:complexType name="JobProblemListType">
        <xs:sequence>
            <xs:element name="problem" type="tns:JobProblemType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*JobProblemDocument*

```
    <xs:element name="JobProblemDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:JobProblemType" />
    <xs:complexType name="JobProblemType">
        <xs:sequence>
            <xs:element name="id" type="xs:long" minOccurs="1" maxOccurs="1"/>
            <xs:element name="type" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="data" type="tns:KeyValueType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="job" type="tns:SiteIdType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="KeyValueType">
        <xs:sequence>
            <xs:element name="key" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="value" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*SearchHistoryDocument*

```
    <xs:element name="SearchHistoryDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SearchHistoryListType" />
    <xs:complexType name="SearchHistoryListType">
        <xs:sequence>
            <xs:element name="search" type="tns:SearchHistoryType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SearchHistoryType">
        <xs:sequence>
            <xs:element name="timestamp" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
            <xs:element name="user" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="query" type="tns:ItemSearchType" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*ImportableFileListDocument*

```
    <xs:element name="ImportableFileListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ImportableFileListType" />
    <xs:complexType name="ImportableFileListType">
        <xs:sequence>
```

```
            <xs:element name="hits" minOccurs="0" maxOccurs="1" type="xs:integer" />
            <xs:element name="element" minOccurs="0" type="tns:ImportableFileType" maxOccurs="unbounded"/>
            <xs:element name="prefixes" type="tns:FilePrefixType" maxOccurs="1" minOccurs="0"></xs:element>
        </xs:sequence>
    </xs:complexType>
```

*ImportableFileDocument*

```
    <xs:element name="ImportableFileDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ImportableFileType" />
    <xs:complexType name="ImportableFileType">
        <xs:sequence>
            <xs:element name="file" type="tns:FileType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="metadata" type="tns:MetadataType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*AutoImportRuleListDocument*

```
    <xs:element name="AutoImportRuleListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:AutoImportRuleListType" />
    <xs:complexType name="AutoImportRuleListType">
        <xs:sequence>
            <xs:element name="rule" type="tns:AutoImportRuleType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*AutoImportRuleDocument*

```
    <xs:element name="AutoImportRuleDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:AutoImportRuleType" />
    <xs:complexType name="AutoImportRuleType">
        <xs:sequence>
            <xs:element name="fileNameAsTitle" type="xs:boolean" minOccurs="0" maxOccurs="1" />
            <xs:element name="storage" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="tag" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="metadata" type="tns:MetadataType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="jobmetadata" type="tns:SimpleMetadataType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="settingsId" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="projection" type="xs:string" minOccurs="0" maxOccurs="1" />
            <xs:element name="excludeFilter" type="tns:FilenameFilterType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="shapeTagFilter" type="tns:FilenameFilterType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="sequenceDefinition" type="tns:sequenceDefinitionType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="priority" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

```
            <xs:element name="user" type="xs:string" minOccurs="0" maxOccurs="1" />
            <xs:element name="ignoreSidecarImport" type="xs:boolean" minOccurs="0" maxOccurs="1" />
            <xs:element name="disabledSidecarExtensions" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="enabled" type="xs:boolean" minOccurs="0" maxOccurs="1" />
            <xs:element name="resourceId" type="tns:SiteIdType" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="sequenceDefinitionType">
        <xs:sequence>
            <xs:element name="sequenceMetadata" type="tns:SequenceMetaDataType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="fileSequence" type="tns:FileSequestionDefinitionType" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SequenceMetaDataType">
        <xs:sequence>
            <xs:element name="regex" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="idGroup" type="xs:integer" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FileSequestionDefinitionType">
        <xs:sequence>
            <xs:element name="regex" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="idGroup" type="xs:integer" minOccurs="1" maxOccurs="1"/>
            <xs:element name="numGroup" type="xs:integer" minOccurs="1" maxOccurs="1"/>
            <xs:element name="timeout" type="xs:integer" minOccurs="1" maxOccurs="1"/>
            <xs:element name="numFormat" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FilenameFilterType">
        <xs:sequence>
            <xs:element name="pattern" type="xs:string"/>
            <xs:element name="tag" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="WeekDayType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="MONDAY"/>
            <xs:enumeration value="TUESDAY"/>
            <xs:enumeration value="WEDNESDAY"/>
            <xs:enumeration value="THURSDAY"/>
            <xs:enumeration value="FRIDAY"/>
            <xs:enumeration value="SATURDAY"/>
            <xs:enumeration value="SUNDAY"/>
        </xs:restriction>
```

```
    </xs:simpleType>
```

*FileSynchronizationInfoDocument*

```
    <xs:element name="FileSynchronizationInfoDocument" type="tns:FileSynchronizationInfoType"/>
    <xs:complexType name="FileSynchronizationInfoType">
        <xs:sequence>
            <xs:element name="id" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="lastUpdated" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
            <xs:element name="size" type="xs:long" minOccurs="1" maxOccurs="1"/>
            <xs:element name="state" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="hash" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FileSynchronizationScheduleEntryType">
        <xs:sequence>
            <xs:element name="day" type="tns:WeekDayType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="start" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="end" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FileSynchronizationScheduleType">
        <xs:sequence>
            <xs:element name="entry" type="tns:FileSynchronizationScheduleEntryType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FileSynchronizationUriMethodType">
        <xs:sequence>
            <xs:element name="scheme" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="methodType" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FileSynchronizationCustomMethodType">
        <xs:sequence>
            <xs:element name="bean" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="additionalParameters" type="tns:SimpleMetadataType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FileSynchronizationMethodType">
        <xs:choice>
            <xs:element name="uri" type="tns:FileSynchronizationUriMethodType"/>
            <xs:element name="custom" type="tns:FileSynchronizationCustomMethodType"/>
        </xs:choice>
    </xs:complexType>
```

*FileSynchronizationEntryListDocument*

```
    <xs:element name="FileSynchronizationEntryListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:FileSynchronizationEntryListType"/>
    <xs:complexType name="FileSynchronizationEntryListType">
        <xs:sequence>
            <xs:element name="entry" type="tns:FileSynchronizationEntryType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FileSynchronizationEntryStatusType">
        <xs:sequence>
            <xs:element name="bytesWritten" type="xs:long" minOccurs="1" maxOccurs="1"/>
            <xs:element name="lastWritten" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="lastChecked" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FileSynchronizationLogEntryType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="timestamp" type="xs:dateTime" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
```

*FileSynchronizationLogDocument*

```
    <xs:element name="FileSynchronizationLogDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:FileSynchronizationLogType"/>
    <xs:complexType name="FileSynchronizationLogType">
        <xs:sequence>
            <xs:element name="entry" type="tns:FileSynchronizationLogEntryType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*FileSynchronizationEntryDocument*

```
    <xs:element name="FileSynchronizationEntryDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:FileSynchronizationEntryType"/>
    <xs:complexType name="FileSynchronizationEntryType">
        <xs:sequence>
            <xs:element name="fileId" type="tns:SiteIdType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="state" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="size" type="xs:long" minOccurs="0" maxOccurs="1"/>
            <xs:element name="hash" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="sourceSite" type="xs:string" minOccurs="1" maxOccurs="1"/>

            <xs:element name="itemId" type="tns:SiteIdType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="shapeId" type="tns:SiteIdType" minOccurs="1" maxOccurs="1"/>

```

```
            <xs:element name="status" type="tns:FileSynchronizationEntryStatusType" minOccurs="0" maxOccurs="1"/>

            <xs:element name="log" type="tns:FileSynchronizationLogType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*FileSynchronizationRuleListDocument*

```
    <xs:element name="FileSynchronizationRuleListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:FileSynchronizationRuleListType"/>
    <xs:complexType name="FileSynchronizationRuleListType">
        <xs:sequence>
            <xs:element name="rule" type="tns:FileSynchronizationRuleType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*FileSynchronizationRuleDocument*

```
    <xs:element name="FileSynchronizationRuleDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:FileSynchronizationRuleType"/>
    <xs:complexType name="FileSynchronizationRuleType">
        <xs:sequence>
            <xs:element name="tag" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="schedule" type="tns:FileSynchronizationScheduleType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="method" type="tns:FileSynchronizationMethodType" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="SyncPolicyType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ONDEMAND"/>
            <xs:enumeration value="ALWAYS"/>
        </xs:restriction>
    </xs:simpleType>
```

*SiteDefinitionDocument*

```
    <xs:element name="SiteDefinitionDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SiteDefinitionType"/>
    <xs:complexType name="SiteDefinitionType">
        <xs:sequence>
            <xs:element name="url" type="xs:string"/>
            <xs:element name="username" type="xs:string"/>
            <xs:element name="password" type="xs:string"/>
            <xs:element name="syncPolicy" type="tns:SyncPolicyType"/>
        </xs:sequence>
    </xs:complexType>
```

*ChangesetUUIDDocument*

```
    <xs:element name="ChangesetUUIDDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ChangesetUUIDType" />
    <xs:complexType name="ChangesetUUIDType">
```

```
        <xs:sequence>
            <xs:element name="uuid" type="xs:string"/>
            <xs:element name="type" type="xs:string"/>
            <xs:element name="id" type="xs:string"/>
            <xs:element name="origin" type="xs:string"/>
            <xs:element name="timestamp" type="xs:dateTime"/>
        </xs:sequence>
    </xs:complexType>
```

*ChangesetUUIDListDocument*

```
    <xs:element name="ChangesetUUIDListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ChangesetUUIDListType"/>
    <xs:complexType name="ChangesetUUIDListType">
        <xs:sequence>
            <xs:element name="changeset" type="tns:ChangesetUUIDType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*SiteInitializationStatusDocument*

```
    <xs:element name="SiteInitializationStatusDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SiteInitializationStatusType"/>
    <xs:complexType name="SiteInitializationStatusType">
        <xs:sequence>
            <xs:element name="started" type="xs:dateTime"/>
            <xs:element name="itemsProcessed" type="xs:integer"/>
            <xs:element name="collectionsProcessed" type="xs:integer"/>
            <xs:element name="librariesProcessed" type="xs:integer"/>
            <xs:element name="usersProcessed" type="xs:integer"/>
            <xs:element name="groupsProcessed" type="xs:integer"/>
        </xs:sequence>
    </xs:complexType>
```

*EntitySynchronizeDocument*

```
    <xs:element name="EntitySynchronizeDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:EntitySynchronizeType" />
    <xs:complexType name="EntitySynchronizeType">
        <xs:sequence>
            <xs:element name="rule" type="tns:SiteRuleType" minOccurs="0"/>
            <xs:element name="createFileStatuses" type="xs:boolean" minOccurs="0" maxOccurs="1" />
            <xs:element name="timestamp" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
            <xs:element name="type" type="xs:string"/>
            <xs:choice>
                <xs:element name="item" type="tns:ItemSynchronizeType"/>
                <xs:element name="collection" type="tns:CollectionSynchronizeType"/>
                <xs:element name="user" type="tns:UserSynchronizeType"/>
                <xs:element name="group" type="tns:GroupSynchronizeType"/>
                <xs:element name="library" type="tns:LibrarySynchronizeType"/>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ItemSynchronizeType">
```

```
        <xs:sequence>
            <xs:element name="delete" type="xs:boolean" />
            <xs:element name="create" type="xs:boolean" />
            <xs:element name="id" type="tns:SiteIdType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="created" type="xs:dateTime" minOccurs="0" maxOccurs="1"/
↪>
            <xs:element name="complete" type="xs:boolean" minOccurs="0" maxOccurs="1"/
↪>
            <xs:element name="metadata" type="tns:MetadataSynchronizeType" minOccurs=
↪"0" maxOccurs="1"/>
            <xs:element name="shape" type="tns:ShapeSynchronizeType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="targetStorageId" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="file" type="tns:FileSynchronizeType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="access" type="tns:AccessControlSynchronizeType"␣
↪minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="thumbnails" type="tns:ThumbnailsSynchronizeType"␣
↪minOccurs="0" maxOccurs="1"/>
            <xs:element name="partOfCollection" type="tns:SiteIdType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="partOfLibrary" type="tns:SiteIdType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="metadataGroup" type="tns:MetadataFieldGroupType"␣
↪minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CollectionSynchronizeType">
        <xs:sequence>
            <xs:element name="delete" type="xs:boolean" />
            <xs:element name="create" type="xs:boolean" />
            <xs:element name="id" type="tns:SiteIdType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="name" type="xs:string" minOccurs="0"  maxOccurs="1"/>
            <xs:element name="complete" type="xs:boolean" minOccurs="0" maxOccurs="1"/
↪>
            <xs:element name="metadata" type="tns:MetadataSynchronizeType" minOccurs=
↪"0" maxOccurs="1"/>
            <xs:element name="access" type="tns:AccessControlSynchronizeType"␣
↪minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="hasItem" type="tns:HasSubEntityType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="hasLibrary" type="tns:SiteIdType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="hasCollection" type="tns:HasSubEntityType" minOccurs="0
↪" maxOccurs="unbounded"/>
            <xs:element name="partOfCollection" type="tns:SiteIdType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="deletedHasItem" type="tns:SiteIdType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="deletedHasLibrary" type="tns:SiteIdType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="deletedHasCollection" type="tns:SiteIdType" minOccurs="0
↪" maxOccurs="unbounded"/>
            <xs:element name="metadataGroup" type="tns:MetadataFieldGroupType"␣
↪minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
```

```
    </xs:complexType>

    <xs:complexType name="LibrarySynchronizeType">
        <xs:sequence>
            <xs:element name="delete" type="xs:boolean" />
            <xs:element name="create" type="xs:boolean" />
            <xs:element name="id" type="tns:SiteIdType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="complete" type="xs:boolean" minOccurs="0" maxOccurs="1"/
↪>
            <xs:element name="access" type="tns:AccessControlSynchronizeType"␣
↪minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="updateMode" type="xs:string" />
            <xs:element name="updateFrequency" type="xs:string" minOccurs="0" />
            <xs:element name="searchXML" type="xs:string" minOccurs="0" />
            <xs:element name="hasItem" type="tns:HasSubEntityType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="partOfCollection" type="tns:SiteIdType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="HasSubEntityType">
        <xs:sequence>
            <xs:element name="id" type="xs:string"/>
            <xs:element name="metadataId" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="UserSynchronizeType">
        <xs:sequence>
            <xs:element name="delete" type="xs:boolean"/>
            <xs:element name="create" type="xs:boolean"/>
            <xs:element name="user" type="tns:UserType"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="GroupSynchronizeType">
        <xs:sequence>
            <xs:element name="removedUser" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
            <xs:element name="removedGroup" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
            <xs:element name="removedRole" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
            <xs:element name="delete" type="xs:boolean"/>
            <xs:element name="create" type="xs:boolean"/>
            <xs:element name="group" type="tns:GroupType"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="MetadataSynchronizeType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType"/>
            <xs:element name="changeSet" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="tns:SiteIdType" minOccurs="1"␣
↪maxOccurs="1"/>
```

```
                        <xs:element name="metadata" type="tns:MetadataEntryListType"
↪minOccurs="1" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*ThumbnailsSynchronizeDocument*

```
    <xs:element name="ThumbnailsSynchronizeDocument" xmlns:tns="http://xml.vidispine.
↪com/schema/vidispine" type="tns:ThumbnailsSynchronizeType" />
    <xs:complexType name="ThumbnailsSynchronizeType">
        <xs:sequence>
            <xs:element name="thumbnail" type="tns:ThumbnailSynchronizeType"
↪minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ThumbnailSynchronizeType">
        <xs:sequence>
            <xs:element name="operation" type="xs:string"/>
            <xs:element name="timecode" type="xs:string"/>
            <xs:element name="version" type="xs:integer"/>
            <xs:element name="poster" type="xs:boolean"/>
            <xs:element name="image" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="MetadataEntryListType">
        <xs:sequence>
            <xs:element name="entry" type="tns:MetadataEntrySyncType" minOccurs="0"
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="MetadataEntrySyncType">
        <xs:sequence>
            <xs:element name="value" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="id" type="tns:SiteIdType"/>
        <xs:attribute name="start" type="xs:string"/>
        <xs:attribute name="end" type="xs:string"/>
        <xs:attribute name="name" type="xs:string"/>
        <xs:attribute name="parentUuid" type="xs:string"/>
        <xs:attribute name="reference" type="xs:boolean"/>
        <xs:attribute name="removed" type="xs:boolean"/>
        <xs:attribute name="timestamp" type="xs:long"/>
        <xs:attribute name="type" type="xs:string"/>
        <xs:attribute name="user" type="xs:string"/>
        <xs:attribute name="valueUuid" type="xs:string"/>
        <xs:attribute name="version" type="xs:integer"/>
        <xs:attribute name="metadataId" type="tns:SiteIdType"/>
        <xs:attribute name="metadataLeafId" type="tns:SiteIdType"/>
        <xs:attribute name="track" type="xs:string"/>
        <xs:attribute name="language" type="xs:string"/>
    </xs:complexType>

```

```
    <xs:complexType name="ShapeSynchronizeType">
        <xs:sequence>
            <xs:element name="delete" type="xs:boolean"/>
            <xs:element name="create" type="xs:boolean"/>
            <xs:element name="essenceVersion" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="component" type="tns:ComponentSynchronizeType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="tag" type="tns:ShapeTagSynchronizeType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="mimeType" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="ComponentTypeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="AUDIO_COMPONENT"/>
            <xs:enumeration value="VIDEO_COMPONENT"/>
            <xs:enumeration value="CONTAINER_COMPONENT"/>
            <xs:enumeration value="BINARY_COMPONENT"/>
            <xs:enumeration value="DESCRIPTOR_COMPONENT"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="ComponentSynchronizeType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="file" type="tns:SiteIdType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="format" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="type" type="tns:ComponentTypeType" minOccurs="1" maxOccurs="1"/>
            <xs:choice>
                <xs:element name="audio" type="tns:AudioComponentType" minOccurs="0" maxOccurs="1"/>
                <xs:element name="container" type="tns:ContainerComponentType" minOccurs="0" maxOccurs="1"/>
                <xs:element name="video" type="tns:VideoComponentType" minOccurs="0" maxOccurs="1"/>
                <xs:element name="binary" type="tns:BinaryComponentType" minOccurs="0" maxOccurs="1"/>
                <xs:element name="descriptor" type="tns:DescriptorComponentType" minOccurs="0" maxOccurs="1"/>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ShapeTagSynchronizeType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="preset" type="tns:TranscodePresetType" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AccessControlSynchronizeType">
```

```
        <xs:sequence>
            <xs:element name="delete" type="xs:boolean"/>
            <xs:element name="create" type="xs:boolean"/>
            <xs:element name="document" type="tns:AccessControlType"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FileSynchronizeType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="uri" type="xs:string" minOccurs="0" maxOccurs="unbounded
↪"/>
            <xs:element name="path" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*FileSiteAvailabilityDocument*

```
    <xs:element name="FileSiteAvailabilityDocument" type="tns:FileSiteAvailabilityType
↪"/>
    <xs:complexType name="FileSiteAvailabilityType">
        <xs:sequence>
            <xs:element type="xs:string" name="site" minOccurs="0" maxOccurs=
↪"unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*FileListDocument*

```
    <xs:element name="FileListDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:FileListType" />
    <xs:complexType name="FileListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="file" type="tns:FileType" maxOccurs="unbounded"␣
↪minOccurs="0"></xs:element>
            <xs:element name="prefixes" type="tns:FilePrefixType" maxOccurs="1"␣
↪minOccurs="0"></xs:element>
            <xs:element name="nextCursor" type="xs:string" maxOccurs="1" minOccurs="0
↪"></xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FilePrefixType">
        <xs:sequence>
            <xs:element name="prefix" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*TransferListDocument*

```
    <xs:element name="TransferListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:TransferListType"/>
    <xs:complexType name="TransferListType">
        <xs:sequence>
            <xs:element name="transfer" type="tns:TransferType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
```

```
        </xs:sequence>
    </xs:complexType>
```

*TransferDocument*

```
    <xs:element name="TransferDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:TransferType"/>
    <xs:complexType name="TransferType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="state" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="priority" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="transferred" type="xs:long" minOccurs="0" maxOccurs="1"/>
            <xs:element name="fileId" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*TransferLogListDocument*

```
    <xs:element name="TransferLogListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:TransferLogListType"/>
    <xs:complexType name="TransferLogListType">
        <xs:sequence>
            <xs:element name="count" type="xs:long" minOccurs="0" maxOccurs="1"/>
            <xs:element name="transferLogEntry" type="tns:TransferLogEntryType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="TransferLogEntryType">
        <xs:sequence>
            <xs:element name="id" type="xs:long" minOccurs="0" maxOccurs="1"/>
            <xs:element name="referredId" type="xs:long" minOccurs="0" maxOccurs="1"/>
            <xs:element name="timestamp" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="method" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="sourceUri" type="xs:anyURI" minOccurs="0" maxOccurs="1"/>
            <xs:element name="destinationUri" type="xs:anyURI" minOccurs="0" maxOccurs="1"/>
            <xs:element name="sourceStorage" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="destinationStorage" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="sourceFile" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="destinationFile" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="sourceItem" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="destinationItem" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="sourceShape" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="destinationShape" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
```

```
            <xs:element name="job" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="status" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FastStartSettingType">
        <xs:sequence>
            <xs:element name="requireFastStart" type="xs:boolean" minOccurs="0" />
            <xs:element name="analyzeDuration" type="xs:boolean" minOccurs="0" />
            <xs:element name="fastStartDuration" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="tns:TimeCodeType">
                            <xs:attribute name="override" type="xs:boolean" use=
↪"required"/>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*TranscodePresetListDocument*

```
    <xs:element name="TranscodePresetListDocument" xmlns:tns="http://xml.vidispine.
↪com/schema/vidispine" type="tns:TranscodePresetListType"/>
    <xs:complexType name="TranscodePresetListType">
        <xs:sequence>
            <xs:element name="preset" type="tns:TranscodePresetType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*TranscodePresetDocument*

```
    <xs:element name="TranscodePresetDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:TranscodePresetType"/>
    <xs:complexType name="TranscodePresetType">
        <xs:sequence>
            <xs:element name="description" type="xs:string" minOccurs="0"/>
            <xs:element name="name" type="xs:string" minOccurs="0"/>
            <xs:element name="format" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="audio" type="tns:AudioTranscodePresetType" minOccurs="0
↪" maxOccurs="1"/>
            <xs:element name="audioTrack" type="tns:AudioTrackTranscodePresetType"␣
↪minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="video" type="tns:VideoTranscodePresetType" minOccurs="0
↪" maxOccurs="1"/>
            <xs:element name="startTimecode" type="xs:string" minOccurs="0" maxOccurs=
↪"1"/>
            <xs:element name="fastStartSetting" type="tns:FastStartSettingType"␣
↪minOccurs="0" maxOccurs="1"/>
            <xs:element name="thumbnailResolution" type="tns:ResolutionType"␣
↪minOccurs="0" maxOccurs="1"/>
            <xs:element name="thumbnailBackground" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="thumbnailPeriod" type="tns:TimeCodeType" minOccurs="0"␣
↪maxOccurs="1"/>
```

```
1374            <xs:element name="thumbnailPlugin" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
1375            <xs:element name="posterResolution" type="tns:ResolutionType" minOccurs="0
↪" maxOccurs="1"/>
1376            <xs:element name="posterBackground" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
1377            <xs:element name="faceDetect" type="xs:boolean" minOccurs="0" maxOccurs="1
↪"/>
1378            <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0"␣
↪maxOccurs="1" />
1379            <xs:element name="preserveEDL" type="xs:boolean" minOccurs="0" maxOccurs=
↪"1"/>
1380            <xs:element name="addClipName" type="xs:boolean" minOccurs="0" maxOccurs=
↪"1"/>
1381            <xs:element name="overlay" minOccurs="0" maxOccurs="unbounded" type="tns:
↪OverlayType"/>
1382            <xs:element name="textOverlay" minOccurs="0" maxOccurs="unbounded" type=
↪"tns:TextOverlayType"/>
1383            <xs:element name="preferredSourceTag" minOccurs="0" maxOccurs="1" type=
↪"xs:string"/>
1384            <xs:element name="script" type="xs:string" minOccurs="0"/>
1385            <xs:element name="shapeMetadata" type="tns:KeyValuePairType" minOccurs="0
↪" maxOccurs="unbounded"/>
1386            <!-- Controls how the maximum time period that each chunk of samples is␣
↪going to be, only used for output of QuickTime files (MOV/MP4) -->
1387            <xs:element name="maxChunkDuration" type="tns:TimeCodeType" minOccurs="0"␣
↪maxOccurs="1"/>
1388            <xs:element name="demuxerSetting" type="tns:KeyValuePairType" minOccurs="0
↪" maxOccurs="unbounded"/>
1389            <xs:element name="muxerSetting" type="tns:KeyValuePairType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
1390            <xs:element name="sequenceOutput" minOccurs="0" maxOccurs="1">
1391                <xs:complexType>
1392                    <xs:sequence>
1393                        <xs:element name="start" type="xs:int" minOccurs="0"␣
↪maxOccurs="1"/>
1394                        <xs:element name="width" type="xs:int" minOccurs="0"␣
↪maxOccurs="1"/>
1395                    </xs:sequence>
1396                </xs:complexType>
1397            </xs:element>
1398            <xs:element name="mediaconvert" minOccurs="0" maxOccurs="1">
1399                <xs:complexType>
1400                    <xs:sequence>
1401                        <xs:element name="inputSetting" type="xs:string" minOccurs="0
↪" maxOccurs="1"/>
1402                        <xs:element name="outputSetting" type="xs:string" minOccurs="0
↪" maxOccurs="unbounded"/>
1403                        <xs:element name="other" type="xs:string" minOccurs="0"␣
↪maxOccurs="unbounded"/>
1404                    </xs:sequence>
1405                </xs:complexType>
1406            </xs:element>
1407        </xs:sequence>
1408    </xs:complexType>
1409
1410    <xs:complexType name="AudioTranscodePresetType">
1411        <xs:sequence>
```

```
            <xs:element name="codec" minOccurs="0" maxOccurs="1" type="xs:string"/>
            <xs:element name="bitrate" minOccurs="0" maxOccurs="1" type="xs:int"/>
            <xs:element name="framerate" minOccurs="0" maxOccurs="1" type="tns:TimeBaseType"/>
            <xs:element name="channel" minOccurs="0" maxOccurs="unbounded" type="xs:int"/>
            <xs:element name="stream" minOccurs="0" maxOccurs="unbounded" type="xs:int"/>
            <xs:element name="preset" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="noAudio" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="mix" type="tns:AudioTranscodePresetMixType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="otif" type="tns:OtifPresetType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="monoFile" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="allChannel" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="output" type="tns:AudioOutputType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AudioOutputType">
        <xs:sequence>
            <xs:element name="format" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="codec" minOccurs="0" maxOccurs="1" type="xs:string"/>
            <xs:element name="bitrate" minOccurs="0" maxOccurs="1" type="xs:int"/>
            <xs:element name="framerate" minOccurs="0" maxOccurs="1" type="tns:TimeBaseType"/>
            <xs:element name="channel" minOccurs="0" maxOccurs="unbounded" type="xs:int"/>
            <xs:element name="stream" minOccurs="0" maxOccurs="unbounded" type="xs:int"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AudioTranscodePresetMixType">
        <xs:sequence>
            <xs:element name="input" type="tns:AudioTranscodePresetChannelMixType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="silence" type="xs:boolean"/>
    </xs:complexType>

    <xs:complexType name="AudioTranscodePresetChannelMixType">
        <xs:sequence>
            <xs:element name="effect" type="tns:EffectType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:int" use="optional"/>
        <xs:attribute name="stream" type="xs:unsignedShort" use="optional"/>
        <xs:attribute name="channel" type="xs:unsignedShort" use="required"/>
        <xs:attribute name="gain" type="xs:float" use="optional"/>
    </xs:complexType>
```

```
    <xs:complexType name="AudioTrackTranscodePresetType">
        <xs:sequence>
            <xs:element name="codec" minOccurs="0" maxOccurs="1" type="xs:string"/>
            <xs:element name="bitrate" minOccurs="0" maxOccurs="1" type="xs:int"/>
            <xs:element name="framerate" minOccurs="0" maxOccurs="1" type="tns:TimeBaseType"/>
            <xs:element name="channel" minOccurs="0" maxOccurs="unbounded" type="xs:int"/>
            <xs:element name="preset" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="mix" type="tns:AudioTranscodePresetMixType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="VideoTranscodePresetType">
        <xs:sequence>
            <xs:element name="scaling" minOccurs="0" maxOccurs="1" type="tns:ScalingType"/>
            <xs:element name="codec" minOccurs="0" maxOccurs="1" type="xs:string"/>
            <xs:element name="bitrate" minOccurs="0" maxOccurs="1" type="xs:int"/>
            <xs:element name="framerate" minOccurs="0" maxOccurs="1" type="tns:TimeBaseType"/>
            <xs:element name="resolution" minOccurs="0" maxOccurs="1" type="tns:ResolutionType"/>

            <xs:element name="displayWidth" type="tns:RationalType" minOccurs="0"/>
            <xs:element name="displayHeight" type="tns:RationalType" minOccurs="0"/>
            <xs:element name="displayXOffset" type="tns:RationalType" minOccurs="0"/>
            <xs:element name="displayYOffset" type="tns:RationalType" minOccurs="0"/>
            <xs:element name="containerSAR" type="tns:AspectRatioType" minOccurs="0"/>

            <xs:element name="forceCFR" minOccurs="0" maxOccurs="1" type="xs:boolean"/>
            <xs:element name="gopSize" type="xs:int" minOccurs="0"/>
            <xs:element name="maxBFrames" type="xs:int" minOccurs="0"/>
            <xs:element name="pixelFormat" type="xs:string" minOccurs="0" />
            <xs:element name="preset" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="profile" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="noVideo" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="stripParameterSets" type="xs:boolean" minOccurs="0"/>
            <xs:element name="addParameterSets" type="xs:boolean" minOccurs="0"/>
            <xs:element name="parameterSets" type="xs:hexBinary" minOccurs="0"/>
            <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="burnTimecode" type="xs:boolean" minOccurs="0"/>
            <xs:element name="burnSubtitles" type="xs:boolean" minOccurs="0"/>
            <xs:element name="imageQuality" type="xs:integer" minOccurs="0"/>
            <xs:element name="otif" type="tns:OtifPresetType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

```
    <xs:simpleType name="OtifPluginType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="audio"/>
            <xs:enumeration value="video"/>
        </xs:restriction>
    </xs:simpleType>
```

*OtifPresetDocument*

```
    <xs:element name="OtifPresetDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:OtifPresetType"/>
    <xs:complexType name="OtifPresetType">
        <xs:sequence>
            <xs:element name="uuid" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="versionMajor" type="xs:int" minOccurs="1" maxOccurs="1"/>
            <xs:element name="versionMinor" type="xs:int" minOccurs="1" maxOccurs="1"/>
            <xs:element name="versionPatch" type="xs:int" minOccurs="1" maxOccurs="1"/>
            <xs:element name="configuration" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="resource" type="tns:NameURIPairType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*OtifConfigurationDocument*

```
    <xs:element name="OtifConfigurationDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:OtifConfigurationType"/>
    <xs:complexType name="OtifConfigurationType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="title" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="preset" type="tns:OtifPresetType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="instance" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*OtifConfigurationListDocument*

```
    <xs:element name="OtifConfigurationListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:OtifConfigurationListType"/>
    <xs:complexType name="OtifConfigurationListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="configuration" type="tns:OtifConfigurationType" maxOccurs="unbounded" minOccurs="0"></xs:element>
        </xs:sequence>
    </xs:complexType>
```

*OtifJobConfigurationDocument*

```
    <xs:element name="OtifJobConfigurationDocument" xmlns:tns="http://xml.vidispine.
↪com/schema/vidispine" type="tns:OtifJobConfigurationType"/>
    <xs:complexType name="OtifJobConfigurationType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0"/>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="configurations" type="tns:OtifConfigurationListType"/>
            <xs:element name="vxa" type="tns:VXAType" minOccurs="1" maxOccurs=
↪"unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*OtifJobConfigurationListDocument*

```
    <xs:element name="OtifJobConfigurationListDocument" xmlns:tns="http://xml.
↪vidispine.com/schema/vidispine" type="tns:OtifJobConfigurationListType"/>
    <xs:complexType name="OtifJobConfigurationListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1" />
            <xs:element name="configuration" type="tns:OtifJobConfigurationType"␣
↪minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
```

*OtifResourceDocument*

```
    <xs:element name="OtifResourceDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:OtifResourceType"/>
    <xs:complexType name="OtifResourceType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="size" type="xs:int" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*OtifResourceListDocument*

```
    <xs:element name="OtifResourceListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:OtifResourceListType"/>
    <xs:complexType name="OtifResourceListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="resource" type="tns:OtifResourceType" maxOccurs=
↪"unbounded" minOccurs="0"></xs:element>
        </xs:sequence>
    </xs:complexType>
```

*StorageRulesDocument*

```
    <xs:element name="StorageRulesDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:StorageRulesType"/>
```

*StorageRuleDocument*

```
    <xs:element name="StorageRuleDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:StorageRuleType"/>

    <xs:complexType name="StorageRulesType">
```

```
        <xs:sequence>
            <xs:element name="default" type="tns:StorageRuleType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="tag" minOccurs="0" maxOccurs="unbounded" type="tns:StorageRuleType"/>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="StorageCriteriaType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="bandwidth"/>
            <xs:enumeration value="capacity"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="StorageRuleType">
        <xs:sequence>
            <xs:element name="storageCount" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="priority" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="tns:StorageCriteriaType">
                            <xs:attribute name="level" type="xs:integer" use="required"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="inherited" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="storage" type="tns:SiteIdType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="group" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="not" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="storage" type="tns:SiteIdType" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="group" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="any" type="tns:EmptyString" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="pool" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="storage" type="tns:SiteIdType" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="group" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="appliesTo" minOccurs="0" maxOccurs="1">
```

```
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="precedence" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" use="optional"/>
    </xs:complexType>

    <xs:simpleType name="EmptyString">
        <xs:restriction base="xs:string">
            <xs:length value="0"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="IntegerOrEmpty">
        <!--xs:union memberTypes="xs:int tns:EmptyString"/-->
        <xs:restriction base="xs:string">
            <xs:pattern value="[0-9]{0,40}"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- START GENERIC ITEM TYPES -->
```

*ItemDocument*

```
    <xs:element name="ItemDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ItemType"/>
    <xs:complexType name="ItemType">
        <xs:sequence>
            <xs:element name="metadata" type="tns:MetadataType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="thumbnails" type="tns:URIListType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="posters" type="tns:URIListType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="files" type="tns:URIListType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="terse" type="tns:GenericType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="shape" type="tns:ShapeType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="merged-access" type="tns:AccessControlMergedType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="access" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="type" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="permission" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    </xs:sequence>
```

```
                </xs:complexType>
            </xs:element>
            <xs:element name="timespan" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="field" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
                                    <xs:element name="value" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                    <xs:attribute name="start" type="xs:string" use="required"/>
                    <xs:attribute name="end" type="xs:string" use="required"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="externalId" type="tns:ExternalIdentifierType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="tns:SiteIdType" use="optional"/>
        <xs:attribute name="start" type="xs:string" use="optional"/>
        <xs:attribute name="end" type="xs:string" use="optional"/>
        <xs:attribute name="base" type="xs:string" use="optional"/>
    </xs:complexType>

    <!--<xs:element name="TerseDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:GenericType"/>
    <xs:complexType name="TestType">
        <xs:sequence>
            <xs:element name="terse" type="vididyn:TerseType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>-->
```

*TerseMetadataDocument*

```
    <xs:element name="TerseMetadataDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:GenericType"/>
```

*TerseMetadataListDocument*

```
    <xs:element name="TerseMetadataListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:TerseMetadataListType"/>
    <xs:complexType name="TerseMetadataListType">
        <xs:sequence>
            <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="tns:GenericType">
                            <xs:attribute name="id" type="tns:SiteIdType" use="required"/>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
```

```
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="GenericType">
        <xs:sequence>
            <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
        </xs:sequence>
    </xs:complexType>

    <!-- END GENERIC ITEM TYPES -->

<!-- START ACCESS CONTROL TYPES -->
```

*AccessControlListDocument*

```
    <xs:element name="AccessControlListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:AccessControlListType"/>
    <xs:complexType name="AccessControlListType">
        <xs:sequence>
            <xs:element name="access" type="tns:AccessControlType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AppliesToType">
        <xs:simpleContent>
            <xs:restriction base="tns:AppliesToValueType">
                <xs:enumeration value="all"/>
                <xs:enumeration value="self"/>
                <xs:enumeration value="collection"/>
                <xs:enumeration value="library"/>
                <xs:enumeration value="item"/>
            </xs:restriction>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="AppliesToValueType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="recursive" type="xs:boolean"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
```

*AccessControlDocument*

```
    <xs:element name="AccessControlDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:AccessControlType"/>
    <xs:complexType name="AccessControlType">
        <xs:sequence>
            <xs:element name="loc" type="xs:anyURI" minOccurs="0" />
            <xs:element name="grantor" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="recursive" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="appliesTo" type="tns:AppliesToType" minOccurs="0" maxOccurs="unbounded"/>
```

```
            <xs:element name="permission" type="xs:string" minOccurs="1" maxOccurs="1
↪"/>
            <xs:element name="priority" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="operation" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:choice>
                        <xs:element name="metadata" type="tns:
↪AccessControlMetadataType" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="shape" type="tns:AccessControlShapeType"␣
↪minOccurs="1" maxOccurs="1"/>
                        <xs:element name="uri" type="tns:AccessControlUriType"␣
↪minOccurs="1" maxOccurs="1"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:choice>
                <xs:element name="group" type="xs:string" minOccurs="1" maxOccurs="1"/
↪>
                <xs:element name="groupType" type="tns:GroupType" minOccurs="1"␣
↪maxOccurs="1"/>
                <xs:element name="user" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="userType" type="tns:UserType" minOccurs="1"␣
↪maxOccurs="1"/>
            </xs:choice>
        </xs:sequence>
        <xs:attribute name="id" type="tns:SiteIdType"/>
    </xs:complexType>

    <xs:complexType name="AccessControlUriType">
        <xs:sequence>
            <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AccessControlShapeType">
        <xs:sequence>
            <xs:element name="tag" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AccessControlMetadataType">
        <xs:sequence>
            <xs:element name="field" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
    <!-- END ACCESS CONTROL TYPES -->
```

*TaskDefinitionListDocument*

```
    <xs:element name="TaskDefinitionListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:TaskDefinitionListType"/>
    <xs:complexType name="TaskDefinitionListType">
        <xs:sequence>
            <xs:element name="task" type="tns:TaskDefinitionType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0"␣
↪maxOccurs="1"/>
        </xs:sequence>
```

```
    </xs:complexType>

    <xs:complexType name="TaskDefinitionDependency">
        <xs:sequence>
            <xs:element name="step" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="previous" type="xs:boolean" minOccurs="0" maxOccurs="1"/
↪>
            <xs:element name="allPrevious" type="xs:boolean" minOccurs="0" maxOccurs=
↪"1"/>
        </xs:sequence>
    </xs:complexType>
```

*TaskDefinitionDocument*

```
    <xs:element name="TaskDefinitionDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:TaskDefinitionType"/>
    <xs:complexType name="TaskDefinitionType">
        <xs:sequence>
            <!-- Optional -->
            <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1
↪"/>
            <xs:element name="extradata" type="xs:string" minOccurs="0" maxOccurs="1"/
↪>
            <xs:element name="flags" type="xs:integer" minOccurs="0" maxOccurs="1"/>

            <!-- Required -->
            <xs:choice>
                <xs:sequence>
                    <xs:element name="bean" type="xs:string" minOccurs="1" maxOccurs=
↪"1"/> <!-- required if not a script step -->
                    <xs:element name="method" type="xs:string" minOccurs="1"␣
↪maxOccurs="1"/> <!-- required if not a script step -->
                    <xs:element name="plugin" type="xs:boolean" minOccurs="0"␣
↪maxOccurs="1"/> <!-- default false -->
                </xs:sequence>
                <xs:element name="script" type="xs:string" minOccurs="1" maxOccurs="1
↪"/>
                <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0
↪" maxOccurs="1"/>
            </xs:choice>
            <xs:element name="step" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="dependency" type="tns:TaskDefinitionDependency"␣
↪minOccurs="0" maxOccurs="1"/>
            <xs:element name="parallelDependency" type="tns:TaskDefinitionDependency"␣
↪minOccurs="0" maxOccurs="1"/>
            <xs:element name="jobType" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="cleanup" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="critical" type="xs:boolean" minOccurs="0" maxOccurs="1"/
↪><!-- default critical -->
        </xs:sequence>
        <xs:attribute name="id" type="xs:integer" use="optional"/>

    </xs:complexType>



    <!-- START NOTIFICATION TYPES -->
```

*NotificationDocument*

```
    <xs:element name="NotificationDocument" type="tns:NotificationType" xmlns:tns="http://xml.vidispine.com/schema/vidispine"/>
    <xs:complexType name="NotificationType">
        <xs:sequence>
            <xs:element name="action" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                    <xs:choice>
                        <xs:element name="http" type="tns:NotificationHttpActionType"/>
                        <xs:element name="ejb" type="tns:NotificationEjbActionType"/>
                        <xs:element name="jms" type="tns:NotificationJmsActionType"/>
                        <xs:element name="javascript" type="tns:NotificationJavaScriptActionType"/>
                        <xs:element name="sqs" type="tns:NotificationSQSActionType"/>
                        <xs:element name="sns" type="tns:NotificationSNSActionType"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="trigger" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                    <xs:choice>
                        <xs:element name="job" type="tns:NotificationJobTriggerType"/>
                        <xs:element name="metadata" type="tns:NotificationMetadataTriggerType"/>
                        <xs:element name="item" type="tns:NotificationItemTriggerType"/>
                        <xs:element name="collection" type="tns:NotificationCollectionTriggerType"/>
                        <xs:element name="storage" type="tns:NotificationStorageTriggerType"/>
                        <xs:element name="file" type="tns:NotificationFileTriggerType"/>
                        <xs:element name="group" type="tns:NotificationGroupTriggerType"/>
                        <xs:element name="access" type="tns:NotificationAccessTriggerType"/>
                        <xs:element name="shape" type="tns:NotificationShapeTriggerType"/>
                        <xs:element name="quota" type="tns:NotificationQuotaTriggerType"/>
                        <xs:element name="document" type="tns:NotificationDocumentTriggerType"/>
                        <xs:element name="deletionLock" type="tns:NotificationDeletionLockTriggerType"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <!-- START NOTIFICATION ACTION TYPES -->
    <xs:complexType name="NotificationActionType">
        <xs:sequence>
            <xs:element name="extradata" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="retry" type="xs:integer"  minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
```

```
        <xs:attribute name="synchronous" type="xs:boolean" use="required"/>
        <xs:attribute name="group" type="xs:string" use="optional"/>
    </xs:complexType>
    <xs:complexType name="NotificationHttpActionType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationActionType">
                <xs:sequence>
                    <xs:element name="contentType" type="xs:string" minOccurs="0"
↪maxOccurs="1"/> <!-- application/xml, application/json, text/plain -->
                    <xs:element name="url" type="xs:string" maxOccurs="1" minOccurs="1
↪"/>
                    <xs:element name="method" type="xs:string" maxOccurs="1"
↪minOccurs="0"/> <!-- defaults to GET -->
                    <xs:element name="timeout" type="xs:string" maxOccurs="1"
↪minOccurs="1"/> <!-- either seconds or "none" -->
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="NotificationJmsActionType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationActionType">
                <xs:sequence>
                    <xs:element name="contentType" type="xs:string" minOccurs="0"
↪maxOccurs="1"/> <!-- application/xml, application/x-java-serialized-object
↪(default) -->
                    <xs:element name="queueFactory" type="xs:string" maxOccurs="1"
↪minOccurs="1"/>
                    <xs:element name="queue" type="xs:string" maxOccurs="1" minOccurs=
↪"1"/>
                    <xs:element name="username" type="xs:string" maxOccurs="1"
↪minOccurs="0"/>
                    <xs:element name="password" type="xs:string" maxOccurs="1"
↪minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="NotificationEjbActionType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationActionType">
                <xs:sequence>
                    <xs:element name="bean" type="xs:string" maxOccurs="1" minOccurs=
↪"1"/>
                    <xs:element name="method" type="xs:string" maxOccurs="1"
↪minOccurs="1"/>
                    <xs:element name="data" type="tns:KeyValuePairType" minOccurs="0"
↪maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="NotificationJavaScriptActionType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationActionType">
                <xs:sequence>
                    <xs:element name="script" type="xs:string" maxOccurs="1"
↪minOccurs="1"/>
```

```
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="NotificationSQSActionType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationActionType">
                <xs:sequence>
                    <xs:element name="contentType" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="endpoint" type="xs:string" maxOccurs="1" minOccurs="1"/>
                    <xs:element name="queue" type="xs:string" maxOccurs="1" minOccurs="1"/>
                    <xs:element name="secret" type="xs:string" maxOccurs="1" minOccurs="0"/>
                    <xs:element name="accessKey" type="xs:string" maxOccurs="1" minOccurs="0"/>
                    <xs:element name="roleArn" type="xs:string" maxOccurs="1" minOccurs="0"/>
                    <xs:element name="roleExternalId" type="xs:string" maxOccurs="1" minOccurs="0"/>
                    <xs:element name="roleSessionName" type="xs:string" maxOccurs="1" minOccurs="0"/>
                    <xs:element name = "messageGroupId" type="xs:string" maxOccurs="1" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="NotificationSNSActionType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationActionType">
                <xs:sequence>
                    <xs:element name="contentType" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="endpoint" type="xs:string" maxOccurs="1" minOccurs="1"/>
                    <xs:element name="topic" type="xs:string" maxOccurs="1" minOccurs="1"/>
                    <xs:element name="secret" type="xs:string" maxOccurs="1" minOccurs="0"/>
                    <xs:element name="accessKey" type="xs:string" maxOccurs="1" minOccurs="0"/>
                    <xs:element name="roleArn" type="xs:string" maxOccurs="1" minOccurs="0"/>
                    <xs:element name="roleExternalId" type="xs:string" maxOccurs="1" minOccurs="0"/>
                    <xs:element name="roleSessionName" type="xs:string" maxOccurs="1" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <!-- END NOTIFICATION ACTION TYPES -->

```

```
    <!-- START NOTIFICATION TRIGGER TYPES -->
```

*NotificationTriggerDocument*

```
    <xs:element name="NotificationTriggerDocument" type="tns:NotificationTriggerType" xmlns:tns="http://xml.vidispine.com/schema/vidispine"/>

    <xs:complexType name="NotificationTriggerType">
        <xs:sequence>
            <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"/> <!- type, e.g. job -->
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="NotificationJobTriggerType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationTriggerType">
                <xs:sequence>
                    <xs:choice>
                        <xs:element name="update" type="xs:string"/>
                        <xs:element name="stop" type="xs:string"/>
                        <xs:element name="finished" type="xs:string"/>
                        <xs:element name="fail" type="xs:string"/>
                        <xs:element name="create" type="xs:string"/>
                    </xs:choice>
                    <xs:element type="xs:boolean" name="placeholder" minOccurs="0" maxOccurs="1"/>

                    <xs:element name="contentFilters" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="contentFilter" minOccurs="0" maxOccurs="unbounded" type="tns:jobNotificationContentFilter"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>

                    <xs:element name="filter" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"/>
                                <xs:element name="step" type="xs:int" minOccurs="0" maxOccurs="1"/>
                                <xs:element name="jobdata" minOccurs="0" maxOccurs="1">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:choice>
                                                <xs:element name="key" type="xs:string" />
                                                <xs:element name="key-regex" type="xs:string" />
                                            </xs:choice>
                                            <xs:choice>
                                                <xs:element name="value" type="xs:string" />
                                                <xs:element name="value-regex" type="xs:string" />
```

```
                                        </xs:choice>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:simpleType name="jobNotificationContentFilter">
    <xs:restriction base="xs:string">
        <xs:enumeration value="jobId"/>
        <xs:enumeration value="jobState"/>
        <xs:enumeration value="user"/>
        <xs:enumeration value="startTime"/>
        <xs:enumeration value="jobType"/>
        <xs:enumeration value="jobData"/>
        <xs:enumeration value="errorMessage"/>
        <xs:enumeration value="itemId"/>
        <xs:enumeration value="totalSteps"/>
        <xs:enumeration value="currentStep"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="NotificationMetadataTriggerType">
    <xs:complexContent>
        <xs:extension base="tns:NotificationTriggerType">
            <xs:choice>
                <xs:element name="modify">
                    <xs:complexType>
                        <xs:sequence>
                            <!-- Unset elements mean "all" -->
                            <xs:element name="field" type="xs:string" minOccurs="0
↪" maxOccurs="1"/>
                            <xs:element name="track" type="xs:string" minOccurs="0
↪" maxOccurs="1"/>
                            <xs:element name="language" type="xs:string"␣
↪minOccurs="0" maxOccurs="1"/>
                            <xs:element name="interval" type="xs:string"␣
↪minOccurs="0" maxOccurs="1"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:choice>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="NotificationItemTriggerType">
    <xs:complexContent>
        <xs:extension base="tns:NotificationTriggerType">
            <xs:choice>
                <xs:element name="modify" type="xs:string"/>
                <xs:element name="delete" type="xs:string"/>
                <xs:element name="create" type="xs:string"/>
            </xs:choice>
```

```
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="NotificationCollectionTriggerType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationTriggerType">
                <xs:choice>
                    <xs:element name="create" type="xs:string"/>
                    <xs:element name="delete" type="xs:string"/>
                    <xs:element name="modify" type="xs:string"/>
                    <xs:element name="item" type="tns:NotificationItemTriggerType"/>
                    <xs:element name="metadata" type="tns:
↪NotificationMetadataTriggerType"/>
                </xs:choice>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="NotificationStorageTriggerType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationTriggerType">
                <xs:choice>
                    <xs:element name="delete" type="xs:string"/>
                    <xs:element name="create" type="xs:string"/>
                    <xs:element name="filename" type="xs:string"/>
                </xs:choice>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="NotificationFileTriggerType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationTriggerType">
                <xs:sequence>
                    <xs:element name="storage" type="tns:SiteIdType" minOccurs="0" />
                    <xs:choice>
                        <xs:element name="new" type="xs:string" />
                        <xs:element name="delete" type="xs:string"/>
                        <xs:element name="change" type="xs:string"/>
                        <xs:element name="hash" type="xs:string"/>
                        <xs:element name="close" type="xs:string"/>
                        <xs:element name="lost" type="xs:string"/>
                    </xs:choice>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="NotificationGroupTriggerType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationTriggerType">
                <xs:choice>
                    <xs:element name="modify" type="xs:string" />
                    <xs:element name="create" type="xs:string"/>
                    <xs:element name="delete" type="xs:string"/>
                </xs:choice>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

```

```
    <xs:complexType name="NotificationAccessTriggerType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationTriggerType">
                <xs:choice>
                    <xs:element name="create" type="xs:string"/>
                    <xs:element name="delete" type="xs:string"/>
                    <xs:element name="change" type="xs:string"/>
                </xs:choice>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="NotificationShapeTriggerType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationTriggerType">
                <xs:choice>
                    <xs:element name="modify" type="xs:string"/>
                    <xs:element name="create" type="xs:string"/>
                    <xs:element name="delete" type="xs:string"/>
                </xs:choice>
           </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="NotificationQuotaTriggerType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationTriggerType">
                <xs:choice>
                    <xs:element name="create" type="xs:string"/>
                    <xs:element name="delete" type="xs:string"/>
                    <xs:element name="warning" type="xs:string"/>
                </xs:choice>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="NotificationDocumentTriggerType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationMetadataTriggerType">
                <xs:choice>
                    <xs:element name="create" type="xs:string"/>
                    <xs:element name="delete" type="xs:string"/>
                </xs:choice>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="NotificationDeletionLockTriggerType">
        <xs:complexContent>
            <xs:extension base="tns:NotificationTriggerType">
                <xs:choice>
                    <xs:element name="create" type="xs:string"/>
                    <xs:element name="delete" type="xs:string"/>
                    <xs:element name="modify" type="xs:string"/>
                    <xs:element name="effective" type="xs:string"/>
                    <xs:element name="expire" type="xs:string"/>
                </xs:choice>
            </xs:extension>
```

```
        </xs:complexContent>
    </xs:complexType>

    <!-- END NOTIFICATION TRIGGER TYPES -->

<!-- END NOTIFICATION TYPES -->
```

```
    <xs:element name ="SupportedProtocolsDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SupportedProtocolsType" />
    <xs:complexType name="SupportedProtocolsType">
        <xs:sequence>
            <xs:element name="source" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="protocol" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="output" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="protocol" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="shape" type="tns:SiteIdType" use="required"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*ItemRelationDocument*

```
    <xs:element name="ItemRelationDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ItemRelationType" />
    <xs:complexType name="ItemRelationType">
        <xs:sequence>
            <xs:element name="id" maxOccurs="1" minOccurs="1" type="xs:string" />
            <xs:element name="direction" maxOccurs="1" minOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="source" type="xs:string" maxOccurs="1" minOccurs="1"/>
                        <xs:element name="target" type="xs:string" maxOccurs="1" minOccurs="1"/>
                    </xs:sequence>
                    <xs:attribute name="type" type="xs:string" use="required"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="value" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attribute name="key" type="xs:string" use="required"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
```

```
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="SortingOrderType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ascending"/>
            <xs:enumeration value="descending"/>
        </xs:restriction>
    </xs:simpleType>
```

*ItemRelationListDocument*

```
    <xs:element name="ItemRelationListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ItemRelationListType" />
    <xs:complexType name="ItemRelationListType">
        <xs:sequence>
            <xs:element name="relation" maxOccurs="unbounded" minOccurs="0" type="tns:ItemRelationType" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ItemSearchValueType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="minimum" type="xs:boolean" use="optional"/>
                <xs:attribute name="maximum" type="xs:boolean" use="optional"/>
                <xs:attribute name="noescape" type="xs:boolean" use="optional"/>
                <xs:attribute name="boost" type="xs:float" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>


    <xs:complexType name="SearchOperatorType">
        <xs:sequence>
            <xs:element name="text" type="tns:ItemSearchTextValueType" maxOccurs="unbounded" minOccurs="0"/>
            <xs:element name="operator" minOccurs="0" maxOccurs="unbounded" type="tns:SearchOperatorType"/>
            <xs:element name="field" minOccurs="0" maxOccurs="unbounded" type="tns:SearchFieldType"/>
            <xs:element name="group" minOccurs="0" maxOccurs="unbounded" type="tns:SearchGroupType"/>
            <xs:element name="reference" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="item" type="tns:ItemCriterionType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="shape" type="tns:ShapeCriterionType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="file" type="tns:CriterionType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="collection" type="tns:CollectionCriterionType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="operation" type="tns:SearchOperationType" use="required"/>
    </xs:complexType>

    <xs:simpleType name="SearchOperationType">
```

```
        <xs:restriction base="xs:string">
            <xs:enumeration value="AND"/>
            <xs:enumeration value="OR"/>
            <xs:enumeration value="NOT"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="SearchFieldType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" maxOccurs="1" minOccurs="1" />
            <xs:element name="value" type="tns:ItemSearchValueType" maxOccurs=
↪"unbounded" minOccurs="0" />
            <xs:element name="range" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="value" type="tns:ItemSearchValueType"␣
↪maxOccurs="2" minOccurs="2" />
                    </xs:sequence>
                    <xs:attribute name="exclusiveMinimum" type="xs:boolean" use=
↪"optional"/>
                    <xs:attribute name="exclusiveMaximum" type="xs:boolean" use=
↪"optional"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="target" type="tns:SearchTargetType"/>
    </xs:complexType>

    <xs:simpleType name="SearchTargetType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="item"/>
            <xs:enumeration value="shape"/>
            <xs:enumeration value="file"/>
        </xs:restriction>
    </xs:simpleType>
```

*MetadataFieldGroupSearchDocument*

```
    <xs:element name="MetadataFieldGroupSearchDocument" xmlns:tns="http://xml.
↪vidispine.com/schema/vidispine" type="tns:ItemSearchType"/>

    <xs:complexType name="SearchGroupType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <!--<xs:element name="referenced" type="tns:MetadataReferencedType"␣
↪minOccurs="0" maxOccurs="1"/>-->
            <xs:element name="operator" type="tns:SearchOperatorType" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:choice>
                <xs:sequence>
                    <xs:element name="field" type="tns:SearchFieldType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
                    <xs:element name="group" type="tns:SearchGroupType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:sequence>
                    <xs:element name="reference" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
```

```
                </xs:sequence>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="SearchIntervalsType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="all"/>
            <xs:enumeration value="generic"/>
            <xs:enumeration value="timed"/>
        </xs:restriction>
    </xs:simpleType>
```

*AutocompleteResponseDocument*

```
    <xs:element name="AutocompleteResponseDocument" xmlns:tns="http://xml.vidispine.
↪com/schema/vidispine" type="tns:AutocompleteResponseType"/>
    <xs:complexType name="AutocompleteResponseType">
        <xs:sequence>
            <xs:element name="field" minOccurs="0" maxOccurs="1" type="xs:string"/>
            <xs:element name="suggestion" minOccurs="0" maxOccurs="unbounded" type=
↪"xs:string"/>
        </xs:sequence>
    </xs:complexType>
```

*AutocompleteRequestDocument*

```
    <xs:element name="AutocompleteRequestDocument" xmlns:tns="http://xml.vidispine.
↪com/schema/vidispine" type="tns:AutocompleteRequestType"/>
    <xs:complexType name="AutocompleteRequestType">
        <xs:sequence>
            <xs:element name="text" minOccurs="1" maxOccurs="1" type="xs:string"/>
        <xs:element name="field" minOccurs="0" maxOccurs="1" type="xs:string"/>
            <xs:element name="maximumSuggestions" minOccurs="0" maxOccurs="1" type=
↪"xs:int"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SuggestionSearchType">
        <xs:sequence>
            <xs:element name="maximumSuggestions" type="xs:int" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="accuracy" type="xs:double" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SuggestionResultType">
        <xs:sequence>
            <xs:element name="term" minOccurs="1" maxOccurs="1" type="xs:string"/>
            <xs:element name="suggestion" minOccurs="0" maxOccurs="unbounded" type=
↪"xs:string"/>
        </xs:sequence>
    </xs:complexType>
```

*GroupSearchDocument*

```
    <xs:element name="GroupSearchDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:SimpleSearchType" />
```

*UserSearchDocument*

```
    <xs:element name="UserSearchDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SimpleSearchType" />
    <xs:complexType name="SimpleSearchType">
        <xs:sequence>
            <xs:element name="sort" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="field" minOccurs="1" maxOccurs="1" type="xs:string"/>
                        <xs:element name="order" minOccurs="1" maxOccurs="1" type="tns:SortingOrderType"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="field" type="tns:SimpleSearchFieldType" maxOccurs="unbounded" minOccurs="0"/>
            <xs:element name="operator" type="tns:SimpleSearchOperatorType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SimpleSearchOperatorType">
        <xs:sequence>
<!--            <xs:element name="operator" minOccurs="0" maxOccurs="unbounded" type="tns:SimpleSearchOperatorType"/>-->
            <xs:element name="field" minOccurs="0" maxOccurs="unbounded" type="tns:SimpleSearchFieldType"/>
        </xs:sequence>
        <xs:attribute name="operation" type="tns:SimpleSearchOperationType" use="required"/>
    </xs:complexType>

    <xs:complexType name="SimpleSearchFieldType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" maxOccurs="1" minOccurs="1" />
            <xs:element name="value" type="xs:string" maxOccurs="1" minOccurs="1" />
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="SimpleSearchOperationType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="AND"/>
            <xs:enumeration value="OR"/>
            <xs:enumeration value="NOT"/>
        </xs:restriction>
    </xs:simpleType>
```

*ShapeSearchDocument*

```
    <xs:element name="ShapeSearchDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ShapeSearchType" />
    <xs:complexType name="ShapeSearchType">
      <xs:complexContent>
        <xs:extension base="tns:ItemSearchType">
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
```

*FileSearchDocument*

```
    <xs:element name="FileSearchDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:FileSearchType" />
    <xs:complexType name="FileSearchType">
      <xs:complexContent>
        <xs:extension base="tns:ItemSearchType">
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ItemSearchTextValueType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="noescape" type="xs:boolean" use="optional"/>
                <xs:attribute name="boost" type="xs:float" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="SearchFilterType">
      <xs:sequence>
        <xs:element name="operator" minOccurs="0" maxOccurs="unbounded" type="tns:SearchOperatorType"/>
        <xs:element name="field" minOccurs="0" maxOccurs="unbounded" type="tns:SearchFieldType"/>
        <xs:element name="group" minOccurs="0" maxOccurs="unbounded" type="tns:SearchGroupType"/>
        <xs:element name="reference" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="item" type="tns:ItemCriterionType" minOccurs="0" maxOccurs="1"/>
        <xs:element name="shape" type="tns:ShapeCriterionType" minOccurs="0" maxOccurs="1"/>
        <xs:element name="file" type="tns:CriterionType" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="operation" type="tns:SearchOperationType" default="AND"/>
      <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>
```

*ItemSearchDocument*

```
    <xs:element name="ItemSearchDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ItemSearchType" />
    <xs:complexType name="ItemSearchType">
        <xs:sequence>
            <xs:element name="text" type="tns:ItemSearchTextValueType" maxOccurs="unbounded" minOccurs="0"/>
            <xs:element name="field" type="tns:SearchFieldType" maxOccurs="unbounded" minOccurs="0"/>
            <xs:element name="group" type="tns:SearchGroupType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="intervals" type="tns:SearchIntervalsType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="reference" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="operator" type="tns:SearchOperatorType" minOccurs="0" maxOccurs="1"/>
```

```
            <xs:element name="filter" type="tns:SearchFilterType" minOccurs="0" maxOccurs="unbounded"/>

            <xs:element name="collection" type="tns:CollectionCriterionType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="item" type="tns:ItemCriterionType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="shape" type="tns:ShapeCriterionType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="file" type="tns:CriterionType" minOccurs="0" maxOccurs="1"/>

            <xs:element name="facetFilter" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="field" minOccurs="1" maxOccurs="1" type="xs:string" />
                <xs:choice>
                  <xs:element name="range" minOccurs="1" maxOccurs="1" type="tns:FacetRangeType" />
                  <xs:element name="value" minOccurs="1" maxOccurs="1" type="xs:string" />
                </xs:choice>
              </xs:sequence>
            </xs:complexType>
            </xs:element>
              <xs:element name="facet" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="field" minOccurs="1" maxOccurs="1" type="xs:string" />
                    <xs:element name="range" minOccurs="0" maxOccurs="unbounded" type="tns:FacetRangeType" />
                    <xs:element name="exclude" minOccurs="0" maxOccurs="unbounded" type="xs:string" />
                  </xs:sequence>
                  <xs:attribute name="count" default="false" type="xs:boolean" />
                  <xs:attribute name="minCount" type="xs:integer" />
                  <xs:attribute name="maxResults" type="xs:integer" />
                  <xs:attribute name="name" type="xs:string" />
                </xs:complexType>
              </xs:element>
              <xs:element name="sort" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="field" minOccurs="1" maxOccurs="1" type="xs:string" />
                    <xs:element name="order" minOccurs="1" maxOccurs="1" type="tns:SortingOrderType" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="highlight" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="field" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
                    <xs:element name="matchingOnly" type="xs:boolean" minOccurs="0" maxOccurs="1" />
```

```
                    <xs:element name="prefix" type="xs:string" minOccurs="0" maxOccurs="1" />
                    <xs:element name="suffix" type="xs:string" minOccurs="0" maxOccurs="1" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="suggestion" minOccurs="0" maxOccurs="1" type="tns:SuggestionSearchType"/>
              <xs:element name="autocomplete" minOccurs="0" maxOccurs="unbounded" type="tns:AutocompleteRequestType"/>
              <xs:element name="cursor" type="xs:string" maxOccurs="1" minOccurs="0"></xs:element>
          </xs:sequence>
          <xs:attribute name="version" type="xs:int" use="optional"/>
    </xs:complexType>

    <xs:complexType name="CriterionType">
      <xs:sequence>
        <xs:element name="field" type="tns:SearchFieldType" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element name="group" type="tns:SearchGroupType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="operator" type="tns:SearchOperatorType" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CollectionCriterionType">
      <xs:complexContent>
        <xs:extension base="tns:CriterionType">
          <xs:sequence>
            <xs:element name="collection" type="tns:CollectionCriterionType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="item" type="tns:ItemCriterionType" minOccurs="0" maxOccurs="1"/>
          </xs:sequence>
          <xs:attribute name="relation" type="tns:RelationType" use="optional"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ItemCriterionType">
      <xs:complexContent>
        <xs:extension base="tns:CriterionType">
          <xs:sequence>
            <xs:element name="shape" type="tns:ShapeCriterionType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="file" type="tns:CriterionType" minOccurs="0" maxOccurs="1"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ShapeCriterionType">
      <xs:complexContent>
        <xs:extension base="tns:CriterionType">
```

```
          <xs:sequence>
            <xs:element name="file" type="tns:CriterionType" minOccurs="0" maxOccurs="1"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>

    <xs:simpleType name="RelationType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="child"/>
        <xs:enumeration value="parent"/>
        <xs:enumeration value="descendant"/>
        <xs:enumeration value="ancestor"/>
      </xs:restriction>
    </xs:simpleType>
```

*ItemListDocument*

```
    <xs:element name="ItemListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ItemListType" />
    <xs:complexType name="ItemListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="library" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="item" minOccurs="0" maxOccurs="unbounded" type="tns:ItemType"/>
            <xs:element name="facet" minOccurs="0" maxOccurs="unbounded" type="tns:FacetType"/>
            <xs:element name="suggestion" minOccurs="0" maxOccurs="unbounded" type="tns:SuggestionResultType"/>
            <xs:element name="autocomplete" minOccurs="0" maxOccurs="unbounded" type="tns:AutocompleteResponseType"/>
            <xs:element name="nextCursor" type="xs:string" maxOccurs="1" minOccurs="0"></xs:element>
        </xs:sequence>
    </xs:complexType>

          <xs:element name="ShapeListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ShapeListType" />
        <xs:complexType name="ShapeListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="shape" minOccurs="0" maxOccurs="unbounded" type="tns:ShapeType"/>
                    <xs:element name="facet" minOccurs="0" maxOccurs="unbounded" type="tns:FacetType"/>
                    <xs:element name="suggestion" minOccurs="0" maxOccurs="unbounded" type="tns:SuggestionResultType"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FacetType">
        <xs:sequence>
            <xs:element name="field" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="count" minOccurs="0" maxOccurs="unbounded" type="tns:FacetCountType"/>
            <xs:element name="range" minOccurs="0" maxOccurs="unbounded" type="tns:FacetRangeType"/>
```

```
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" />
    </xs:complexType>

    <xs:complexType name="FacetCountType">
        <xs:simpleContent>
            <xs:extension base="xs:long">
                <xs:attribute name="fieldValue" type="xs:string" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="FacetRangeType">
        <xs:simpleContent>
            <xs:extension base="tns:IntegerOrEmpty">
                <xs:attribute name="start" type="xs:string" use="required"/>
                <xs:attribute name="end" type="xs:string" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
```

*MetadataChangeSetDocument*

```
    <xs:element name="MetadataChangeSetDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:MetadataChangeSetType" />


    <xs:complexType name="MetadataChangeSetType">
        <xs:sequence>
            <xs:element name="changeSet" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="tns:SiteIdType" minOccurs="1"␣
↪maxOccurs="1"/>
                        <xs:element name="metadata" type="tns:MetadataType" minOccurs=
↪"1" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*JobListDocument*

```
    <xs:element name="JobListDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:JobListType" />
    <xs:complexType name="JobListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="job" type="tns:JobType" minOccurs="0" maxOccurs=
↪"unbounded"/>
            <xs:element name="facet" type="tns:FacetType" minOccurs="0" maxOccurs=
↪"unbounded"/>
            <xs:element name="notFound" type="tns:NotFoundExceptionType" minOccurs="0
↪" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*JobDocument*

```
    <xs:element name="JobDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:JobType" />
    <xs:complexType name="JobType">
        <xs:sequence>
            <xs:element name="jobId" type="tns:SiteIdType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="user" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="started" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="finished" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="status" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="type" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="subJob" type="tns:JobType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="priority" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="waiting" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="resourceId" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="resourceType" type="xs:string" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="requirement" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="currentStep" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="description" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="number" type="xs:int" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="status" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="data" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="key" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="value" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="totalSteps" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="log" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="task" type="tns:JobTaskType" minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
```

```
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="JobTaskType">
        <xs:sequence>
            <xs:element name="step" type="xs:int" minOccurs="1" maxOccurs="1"/>
            <xs:element name="attempts" type="xs:int" minOccurs="1" maxOccurs="1"/>
            <xs:element name="status" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="timestamp" type="xs:dateTime" minOccurs="1" maxOccurs="1
↪"/>
            <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1
↪"/>
            <xs:element name="progress" type="tns:JobTaskProgressType" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="subStep" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="timestamp" type="xs:dateTime" minOccurs="1"␣
↪maxOccurs="1"/>
                        <xs:element name="description" type="xs:string" minOccurs="1"␣
↪maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="errorMessage" type="xs:string" minOccurs="0" maxOccurs=
↪"1"/>
            <xs:element name="totalSubTasks" type="xs:int" minOccurs="0" maxOccurs="1
↪"/>
            <xs:element name="subTask" type="tns:JobTaskType" minOccurs="0" maxOccurs=
↪"unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:int" use="optional"/>
    </xs:complexType>

    <xs:complexType name="JobTaskProgressType">
        <xs:simpleContent>
            <xs:extension base="xs:decimal">
                <xs:attribute name="total" type="xs:long" use="optional"/>
                <xs:attribute name="unit" type="tns:JobTaskProgressType_unit" use=
↪"optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:simpleType name="JobTaskProgressType_unit">
        <xs:restriction base="xs:string">
            <xs:enumeration value="bytes"/>
            <xs:enumeration value="percent"/>
        </xs:restriction>
    </xs:simpleType>
```

*JobSearchDocument*

```
    <xs:element name="JobSearchDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:JobSearchType" />
    <xs:complexType name="JobSearchType">
```

```
        <xs:sequence>
            <xs:element name="facetFilter" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="field" minOccurs="1" maxOccurs="1" type="xs:string" />
                        <xs:choice>
                            <xs:element name="range" minOccurs="1" maxOccurs="1" type="tns:FacetRangeType" />
                            <xs:element name="value" minOccurs="1" maxOccurs="1" type="xs:string" />
                        </xs:choice>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="facet" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="field" minOccurs="1" maxOccurs="1" type="xs:string" />
                        <xs:element name="exclude" minOccurs="0" maxOccurs="unbounded" type="xs:string" />
                    </xs:sequence>
                    <xs:attribute name="count" default="false" type="xs:boolean" />
                    <xs:attribute name="minCount" default="0" type="xs:integer" />
                    <xs:attribute name="maxCount" default="100" type="xs:integer" />
                    <xs:attribute name="maxResults" default="100" type="xs:integer" />
                    <xs:attribute name="name" type="xs:string" />
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*StorageMethodListDocument*

```
    <xs:element name="StorageMethodListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:StorageMethodListType"/>
    <xs:complexType name="StorageMethodListType">
        <xs:sequence>
            <xs:element name="method" type="tns:StorageMethodType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="StorageMethodType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="loc"  type="xs:anyURI" minOccurs="0"/>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0"/>
            <xs:element name="uri"  type="xs:anyURI"/>
            <xs:element name="bandwidth" minOccurs="0" type="xs:long"/>
            <xs:element name="read" minOccurs="1" type="xs:boolean"/>
            <xs:element name="write" minOccurs="1" type="xs:boolean"/>
            <xs:element name="browse" minOccurs="1" type="xs:boolean"/>
            <xs:element name="lastSuccess" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="lastFailure" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="failureMessage" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

```
            <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0"
↪maxOccurs="1" />
        </xs:sequence>
    </xs:complexType>
```

*StorageDocument*

```
    <xs:element name="StorageDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:StorageType" />
    <xs:complexType name="StorageType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="state" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="priority" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="capacity" type="xs:long" minOccurs="0" maxOccurs="1"/>
            <xs:element name="freeCapacity" minOccurs="0" type="xs:long"/>
            <xs:element name="bandwidth" minOccurs="0" type="xs:long"/>
            <xs:element name="timestamp" minOccurs="0" type="xs:dateTime"/>
            <xs:element name="method" type="tns:StorageMethodType" maxOccurs=
↪"unbounded" minOccurs="0"/>
            <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0"
↪maxOccurs="1" />
            <xs:element name="lowWatermark" type="xs:long" minOccurs="0" maxOccurs="1
↪"/>
            <xs:element name="highWatermark" type="xs:long" minOccurs="0" maxOccurs="1
↪"/>
            <xs:element name="lowWatermarkPercentage" type="xs:int" minOccurs="0"
↪maxOccurs="1"/>
            <xs:element name="highWatermarkPercentage" type="xs:int" minOccurs="0"
↪maxOccurs="1"/>
            <xs:element name="autoDetect" type="xs:boolean" minOccurs="0" maxOccurs="1
↪"/>
            <xs:element name="bean" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="showImportables" type="xs:boolean" minOccurs="0"
↪maxOccurs="1"/>
            <xs:element name="projection" type="xs:string" minOccurs="0" maxOccurs="1
↪"/>
            <xs:element name="scanInterval" type="xs:int" minOccurs="0" maxOccurs="1"/
↪>
            <xs:element name="archiveScript" type="xs:string" minOccurs="0" maxOccurs=
↪"1"/>
            <xs:element name="sequence" type="tns:StorageFileSequenceType" minOccurs=
↪"0" maxOccurs="unbounded"/>
            <xs:element name="sequenceTimeout" type="xs:int" minOccurs="0" maxOccurs=
↪"1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="StorageFileSequenceType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="regex" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="numGroup" type="xs:int" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*StorageListDocument*

```
    <xs:element name="StorageListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:StorageListType"/>
    <xs:complexType name="StorageListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="storage" type="tns:StorageType" maxOccurs="unbounded" minOccurs="0"></xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="VXAStatus">
        <xs:restriction base="xs:string">
            <xs:enumeration value="OFFLINE"/>
            <xs:enumeration value="ONLINE"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="VXAStorageType">
        <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="name" type="xs:string" minOccurs="1"/>
            <xs:element name="id" type="xs:string" minOccurs="1"/>
            <xs:element name="path" type="xs:string" minOccurs="1"/>
            <xs:element name="isCollectionStorage" type="xs:string" minOccurs="0"/>
            <xs:element name="createProxiesStorage" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="VXAVSInstanceType">
        <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="vsClusterAddress" type ="xs:string" minOccurs="0" maxOccurs="1"/> <!-- from VS, status only -->
            <xs:element name="uri" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="status" type="tns:VXAStatus" minOccurs="0" maxOccurs="1"/>
            <xs:element name="lastSeen" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*VXAForwardService*

```
    <xs:element name="VXAForwardService" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ForwardServiceType"/>
    <xs:complexType name="ForwardServiceType">
        <xs:sequence minOccurs="0" maxOccurs="1">
            <xs:element name="id" type="xs:int" minOccurs="1" maxOccurs="1"/>
            <xs:element name="uri" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*VXADocument*

```
    <xs:element name="VXADocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:VXAType" />
    <xs:complexType name="VXAType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="uuid" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="user" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

```
            <xs:element name="allStorages" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="storage" type="tns:VXAStorageType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="file" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="instance" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="vxaVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="s3CredentialType" type="tns:S3CredentialType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="transcoderVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="uri" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="port" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="status" type="tns:VXAStatus" minOccurs="0" maxOccurs="1"/>
            <xs:element name="lastSeen" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="mode" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="publicKey" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="vsClusterAddress" type ="xs:string" minOccurs="0" maxOccurs="1"/> <!-- from VS to VSA -->
            <xs:element name="vsInstance" type ="tns:VXAVSInstanceType" minOccurs="0" maxOccurs="unbounded"/> <!-- from VS, status only -->
            <xs:element name="transcoder" type="tns:TranscoderType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="agentGroup" type="xs:string" minOccurs="0" maxOccurs="1"/> <!-- Agents in the same group are assumed to be able to connect to each other -->
            <xs:element name="externalUri" type="xs:string" minOccurs="0" maxOccurs="1"/> <!-- URI where the VSA can be reached, given the agentGroup -->
            <xs:element name="forwardService" type="tns:ForwardServiceType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="S3CredentialType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="none"/>
            <xs:enumeration value="temporary"/>
            <xs:enumeration value="secretkey"/>
        </xs:restriction>
    </xs:simpleType>
```

*VXAListDocument*

```
    <xs:element name="VXAListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:VXAListType"/>
    <xs:complexType name="VXAListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="vxa" type="tns:VXAType" maxOccurs="unbounded" minOccurs="0"></xs:element>
        </xs:sequence>
    </xs:complexType>
```

```
    <xs:simpleType name="OSName">
        <xs:restriction base="xs:string">
            <xs:enumeration value="DEBIAN32"/>
            <xs:enumeration value="DEBIAN64"/>
            <xs:enumeration value="REDHAT32"/>
            <xs:enumeration value="REDHAT64"/>
            <xs:enumeration value="WINDOWS32"/>
            <xs:enumeration value="WINDOWS64"/>
            <xs:enumeration value="MACOSX32"/>
            <xs:enumeration value="MACOSX64"/>
        </xs:restriction>
    </xs:simpleType>
```

*OtifOSDocment*

```
    <xs:element name="OtifOSDocment" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:OtifOSType"/>
    <xs:complexType name="OtifOSType">
        <xs:sequence>
            <xs:element name="name" type="tns:OSName" minOccurs="1" maxOccurs="1"/>
            <xs:element name="file" type="xs:string" minOccurs="1" maxOccurs=
↪"unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*OtifTranscoderPluginDocment*

```
    <xs:element name="OtifTranscoderPluginDocment" xmlns:tns="http://xml.vidispine.
↪com/schema/vidispine" type="tns:OtifTranscoderPluginType"/>
    <xs:complexType name="OtifTranscoderPluginType">
        <xs:sequence>
            <xs:element name="pluginType" type="tns:OtifPluginType" minOccurs="1"␣
↪maxOccurs="1"/>
            <xs:element name="os" type="tns:OtifOSType" minOccurs="0" maxOccurs=
↪"unbounded"/>
            <xs:element name="file" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*OtifVxaPluginDocment*

```
    <xs:element name="OtifVxaPluginDocment" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:OtifVxaPluginType"/>
    <xs:complexType name="OtifVxaPluginType">
        <xs:sequence>
            <xs:element name="file" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*OtifDocument*

```
    <xs:element name="OtifDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:OtifType" />
    <xs:complexType name="OtifType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <!-- UUID used in presets/analyze/complex-jobs to tell which plugin to␣
↪use -->
```

```
            <xs:element name="uuid" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <!-- pluginName - human readable name of plugin -->
            <xs:element name="pluginName" type="xs:string" minOccurs="1" maxOccurs="1
↪"/>
            <!-- vendorName Plugin vendor name -->
            <xs:element name="vendorName" type="xs:string" minOccurs="1" maxOccurs="1
↪"/>
            <xs:element name="versionMajor" type="xs:int" minOccurs="1" maxOccurs="1"/
↪>
            <xs:element name="versionMinor" type="xs:int" minOccurs="1" maxOccurs="1"/
↪>
            <xs:element name="versionPatch" type="xs:int" minOccurs="1" maxOccurs="1"/
↪>
            <!-- Optional transcoderPlugin, at least one of transcoderPlugin and␣
↪vxaPlugin must be present -->
            <xs:element name="transcoderPlugin" type="tns:OtifTranscoderPluginType"␣
↪minOccurs="0" maxOccurs="1"/>
            <!-- Optional vxaPlugin, at least one of transcoderPlugin and vxaPlugin␣
↪must be present -->
            <xs:element name="vxaPlugin" type="tns:OtifVxaPluginType" minOccurs="0"␣
↪maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*OtifListDocument*

```
    <xs:element name="OtifListDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:OtifListType"/>
    <xs:complexType name="OtifListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="otif" type="tns:OtifType" maxOccurs="unbounded"␣
↪minOccurs="0"></xs:element>
        </xs:sequence>
    </xs:complexType>
```

*VXAStorageCapacityDocument*

```
    <xs:element name="VXAStorageCapacityDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:VXAStorageCapacityType" />
    <xs:complexType name="VXAStorageCapacityType">
        <xs:sequence minOccurs="1" maxOccurs="1">
            <xs:element name="free" type="xs:long" minOccurs="0" maxOccurs="1"/>
            <xs:element name="total" type="xs:long" minOccurs="0" maxOccurs="1"/>
            <xs:element name="used" type="xs:long" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*QuotaRuleListDocument*

```
    <xs:element name="QuotaRuleListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:QuotaRuleListType"/>
    <xs:complexType name="QuotaRuleListType">
        <xs:sequence>
            <xs:element name="rule" type="tns:QuotaRuleType" minOccurs="0" maxOccurs=
↪"unbounded"></xs:element>
        </xs:sequence>
    </xs:complexType>
```

*QuotaRuleDocument*

```
    <xs:element name="QuotaRuleDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:QuotaRuleType"/>
    <xs:complexType name="QuotaRuleType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>

            <!-- Filters -->
            <xs:choice minOccurs="0">
                <xs:element name="user" type="xs:string"/>
                <xs:element name="group" type="xs:string"/>
            </xs:choice>
            <xs:choice minOccurs="0">
                <xs:element name="collection" type="tns:SiteIdType"/>
                <xs:element name="library" type="tns:SiteIdType"/>
            </xs:choice>
            <xs:choice minOccurs="0">
                <xs:element name="storage" type="tns:SiteIdType"/>
                <xs:element name="storageGroup" type="tns:SiteIdType"/>
            </xs:choice>
            <xs:element name="tag" type="xs:string" minOccurs="0" maxOccurs="1"/>

            <!-- Resource Limits -->
            <xs:element name="resource" minOccurs="1" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="name" type="tns:QuotaResourceType" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="limit" type="xs:long" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="usage" type="xs:long" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <!-- Other -->
            <xs:element name="updateFrequency" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="lastUpdate" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="externalId" type="tns:ExternalIdentifierType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="QuotaResourceType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="item"/>
            <xs:enumeration value="storage"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="TranscoderDirectAccess">
        <xs:sequence>
            <xs:element name="filter" type="xs:string" minOccurs="1" maxOccurs="1"/>
```

```
            <xs:element name="rewrite" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element type="xs:string" name="pattern" minOccurs="1" maxOccurs="1"/>
                        <xs:element type="xs:string" name="replacement" minOccurs="1" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="TranscoderType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="type" minOccurs="0" maxOccurs="1" default="TRANSCODER">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="TRANSCODER"/>
                        <xs:enumeration value="DIRECTORY"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="url" type="xs:anyURI"></xs:element>
            <xs:element name="version" type="xs:string" minOccurs="0"></xs:element>
            <xs:element name="reverseAddress" type="xs:string" minOccurs="0"></xs:element>
            <xs:element name="reverseAddressDetected" type="xs:string" minOccurs="0"></xs:element>
            <xs:element name="directAccess" type="tns:TranscoderDirectAccess" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="state" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="job" type="tns:JobStatusType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="configuration" type="tns:TranscoderConfigurationType" minOccurs="0" maxOccurs="1" />
            <xs:element name="transcoder" type="tns:TranscoderType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="weight" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="maxJob" type="xs:int" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FinalCutServerType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="url" type="xs:anyURI"></xs:element>
            <xs:element name="tag" type="xs:string"></xs:element>
            <xs:element name="state" type="xs:string" minOccurs="0" maxOccurs="1" />
            <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0" maxOccurs="1" />
            <xs:element name="description" type="xs:string" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="MXFServerResourceType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="url" type="xs:anyURI"></xs:element>
            <xs:element name="workspaceUrl" type="xs:anyURI"></xs:element>
```

```
3083            <xs:element name="userWorkspaceUrl" type="xs:anyURI"></xs:element>
3084            <xs:element name="mxfServerWorkspacePath" type="xs:string"></xs:element>
3085            <xs:element name="mxfServerUserId" type="xs:integer"></xs:element>
3086            <xs:element name="mxfServerPathToStorage" type="xs:anyURI"></xs:element>
3087            <xs:element name="databaseName" type="xs:string"/>
3088            <xs:element name="storageId" type="tns:SiteIdType"></xs:element>
3089            <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0"
    ↪maxOccurs="1" />
3090            <xs:element name="description" type="xs:string" minOccurs="0" />
3091            <xs:element name="atomShapes" type="xs:string" minOccurs="0" />
3092            <xs:element name="importShapes" type="xs:string" minOccurs="0" />
3093            <xs:element name="detectAtom" type="xs:boolean" minOccurs="0" />
3094            <xs:element name="enforceQuota" type="xs:boolean" minOccurs="0" />
3095            <xs:element name="fileImportPattern" type="xs:string" minOccurs="0" />
3096        </xs:sequence>
3097    </xs:complexType>
3098
3099    <xs:complexType name="SigniantType">
3100        <xs:sequence maxOccurs="1" minOccurs="1">
3101            <xs:element name="tag" type="xs:string" />
3102            <xs:element name="url" type="xs:anyURI" />
3103            <xs:element name="username" type="xs:string" />
3104            <xs:element name="password" type="xs:string" />
3105            <xs:element name="description" type="xs:string" minOccurs="0" />
3106        </xs:sequence>
3107    </xs:complexType>
3108
3109    <xs:complexType name="NetworkType">
3110        <xs:sequence maxOccurs="1" minOccurs="1">
3111            <xs:element name="netmask" type="xs:anyURI"></xs:element>
3112            <xs:element name="bandwidth" minOccurs="0" type="xs:long"/>
3113        </xs:sequence>
3114    </xs:complexType>
3115
3116    <xs:complexType name="ThumbnailServiceType">
3117        <xs:sequence maxOccurs="1" minOccurs="1">
3118            <xs:element name="path" type="xs:string"/>
3119            <xs:element name="state" type="xs:string" minOccurs="1" maxOccurs="1"/>
3120            <xs:element name="lastSuccess" type="xs:dateTime" minOccurs="0" maxOccurs=
    ↪"1"/>
3121            <xs:element name="lastFailure" type="xs:dateTime" minOccurs="0" maxOccurs=
    ↪"1"/>
3122            <xs:element name="failureMessage" type="xs:string" minOccurs="0"
    ↪maxOccurs="1"/>
3123            <xs:element name="mode" type="xs:string" minOccurs="0" maxOccurs="1"/>
3124        </xs:sequence>
3125    </xs:complexType>
3126
3127    <xs:complexType name="LDAPImportType">
3128        <xs:sequence>
3129            <xs:element name="interval" type="xs:long" minOccurs="0" maxOccurs="1"/>
3130            <xs:element name="importOrganizationalUnits" type="xs:boolean" minOccurs=
    ↪"0" maxOccurs="1"/>
3131
3132            <xs:sequence minOccurs="0" maxOccurs="1">
3133                <xs:element name="plugin" type="xs:string" minOccurs="1" maxOccurs="1
    ↪"/>
3134                <xs:element name="pluginParameters" type="tns:SimpleMetadataType"
    ↪minOccurs="1" maxOccurs="1"/>
```

```
            </xs:sequence>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="LDAPSyncType">
        <xs:complexContent>
            <xs:extension base="tns:LDAPImportType">
                <xs:sequence>
                    <xs:element name="createUsers" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="createGroups" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="LDAPResourceType">
        <xs:sequence>
            <!-- Required -->
            <xs:element name="url" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
            <xs:element name="useStartTLS" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
            <xs:element name="userDN" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="usernameAttribute" type="xs:string" minOccurs="1" maxOccurs="1"/>

            <!-- Optional -->
            <xs:element name="userSearchFilter" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="bindDN" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="bindPassword" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="cacheLifetime" type="xs:long" minOccurs="0" maxOccurs="1"/>

            <xs:element name="groupDN" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="groupSearchFilter" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="realNameAttribute" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="groupnameAttribute" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="usernameFormat" type="xs:string" minOccurs="0" maxOccurs="1"/>

      <xs:element name="secureProtocol" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="serverCertificate" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>

        <xs:element name="sync" type="tns:LDAPSyncType" minOccurs="0" maxOccurs="1"/>

        <!-- Deprecated. Use sync instead -->
            <xs:element name="import" type="tns:LDAPImportType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

```
    <xs:complexType name="ExternalTranscoderType">
        <xs:sequence>
            <xs:element name="source" type="xs:string" maxOccurs="1" minOccurs="1"/>
            <xs:element name="destination" type="xs:string" maxOccurs="1" minOccurs="1
↪"/>
            <xs:element name="shapeTag" type="xs:string" maxOccurs="1" minOccurs="1"/>
            <xs:element name="timeout" type="xs:long" maxOccurs="1" minOccurs="0"/>
            <xs:element name="interval" type="xs:long" maxOccurs="1" minOccurs="0"/>
            <xs:element name="checks" type="xs:int" maxOccurs="1" minOccurs="0"/>
            <xs:element name="regex" type="xs:string" maxOccurs="1" minOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CerifyType">
        <xs:sequence>
            <xs:element name="address" type="xs:string" maxOccurs="1" minOccurs="1"/>
            <xs:element name="mediaLocation" maxOccurs="unbounded" minOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="name" type="xs:string" maxOccurs="1"␣
↪minOccurs="1"/>
                        <xs:element name="storageMethod" type="tns:SiteIdType"␣
↪maxOccurs="unbounded" minOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="cleanup" type="xs:boolean" maxOccurs="1" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
```

*ExternalTranscodeJobDocument*

```
    <xs:element name="ExternalTranscodeJobDocument" xmlns:tns="http://xml.vidispine.
↪com/schema/vidispine" type="tns:ExternalTranscodeJobType"/>
    <xs:complexType name="ExternalTranscodeJobType">
        <xs:sequence>
            <xs:element name="sourceUri" type="xs:string" maxOccurs="1" minOccurs="1"/
↪>
            <xs:element name="format" type="xs:string" maxOccurs="1" minOccurs="1"/>
            <xs:element name="transcoder" type="tns:ExternalTranscoderType" maxOccurs=
↪"1" minOccurs="1"/>
            <xs:element name="regex" type="xs:string" maxOccurs="1" minOccurs="1"/>
            <xs:element name="storageId" type="xs:string" maxOccurs="1" minOccurs="1"/
↪>
            <xs:element name="username" type="xs:string" maxOccurs="1" minOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="VidinetServiceType">
        <xs:sequence>
            <xs:element name="url" type="xs:anyURI" maxOccurs="1" minOccurs="1"/>
            <xs:element name="name" type="xs:string" maxOccurs="1" minOccurs="0"/>
            <xs:element name="endpoint" type="xs:anyURI" maxOccurs="1" minOccurs="0"/>
            <xs:element name="type" type="xs:string" maxOccurs="1" minOccurs="1"/>
            <xs:element name="state" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="scheme" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
```

```
        </xs:sequence>
    </xs:complexType>
```

*ResourceDocument*

```
    <xs:element name="ResourceDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ResourceType" />
    <xs:complexType name="ResourceType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0"/>
            <xs:element name="state" type="xs:string" minOccurs="0"/>
            <xs:choice>
                <xs:element name="network" type="tns:NetworkType"></xs:element>
                <xs:element name="transcoder" type="tns:TranscoderType"></xs:element>
                <xs:element name="externalTranscoder" type="tns:ExternalTranscoderType"></xs:element>
                <xs:element name="cerify" type="tns:CerifyType"></xs:element>
                <xs:element name="thumbnail" type="tns:ThumbnailServiceType"></xs:element>
                <xs:element name="finalcutserver" type="tns:FinalCutServerType"></xs:element>
                <xs:element name="mxfserver" type="tns:MXFServerResourceType"></xs:element>
                <xs:element name="signiant" type="tns:SigniantType"></xs:element>
                <xs:element name="ldap" type="tns:LDAPResourceType"/>
                <xs:element name="unknown" type="xs:string"></xs:element>
                <xs:element name="cloudconvert" type="tns:CloudConvertType"></xs:element>
                <xs:element name="vidinet" type="tns:VidinetServiceType"></xs:element>
                <xs:element name="eidr" type="tns:EidrType"></xs:element>
                <xs:element name="callback" type="tns:CallbackLocationResourceType"></xs:element>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CallbackLocationResourceType">
        <xs:sequence>
            <xs:element name="uri" type="xs:anyURI"></xs:element>
        </xs:sequence>
    </xs:complexType>
```

*CallbackDocument*

```
    <xs:element name="CallbackDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:CallbackDocumentType" />
    <xs:complexType name="CallbackDocumentType">
        <xs:sequence>
            <xs:element name="name" type="xs:string"></xs:element>
            <xs:element name="description" type="xs:string"></xs:element>
            <xs:element name="script" type="xs:string"></xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CloudConvertType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="apiKey" type="xs:string"></xs:element>
            <xs:element name="apiHost" type="xs:string"></xs:element>
```

```
            <xs:element name="script" type="xs:string"></xs:element>
            <xs:element name="inputMethod" type="xs:string"></xs:element>
            <xs:element name="publicAddress" type="xs:anyURI"></xs:element>
        </xs:sequence>
    </xs:complexType>
```

*EidrDocument*

```
    <xs:element name="EidrDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:EidrType" />
    <xs:complexType name="EidrType">
        <xs:sequence>
            <xs:element name="url" type="xs:anyURI" maxOccurs="1" minOccurs="1"/>
            <xs:element name="include" type="xs:string" maxOccurs="unbounded"␣
↪minOccurs="0"/> <!-- default:"eidr_base"-->
            <xs:element name="partyId" type="xs:string" maxOccurs="1" minOccurs="0"/>
            <xs:element name="userId" type="xs:string" maxOccurs="1" minOccurs="0"/>
            <xs:element name="password" type="xs:string" maxOccurs="1" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
```

*ResourceListDocument*

```
    <xs:element name="ResourceListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:ResourceListType" />
    <xs:complexType name="ResourceListType">
            <xs:sequence>
                <xs:element name="resource" type="tns:ResourceType" maxOccurs=
↪"unbounded" minOccurs="0"></xs:element>
            </xs:sequence>
    </xs:complexType>
```

*ResourceTypeListDocument*

```
    <xs:element name="ResourceTypeListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:ResourceTypeListType" />
    <xs:complexType name="ResourceTypeListType">
            <xs:sequence>
                <xs:element name="resourcetype" maxOccurs="unbounded" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="type" type="xs:string"></xs:element>
                            <xs:element name="url" type="xs:anyURI" minOccurs="0"></
↪xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
    </xs:complexType>
```

*CostEstimateDocument*

```
    <xs:element name="CostEstimateDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:CostEstimateType" />
    <xs:complexType name="CostEstimateType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0"/>
            <xs:element name="url" type="xs:anyURI" minOccurs="0"/>
```

```
            <xs:element name="state" type="xs:string" minOccurs="0"/>
            <xs:element name="service" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="resource" type="tns:SiteIdType"/>
                        <xs:element name="name" type="xs:string" minOccurs="0"/>
                        <xs:element name="type" type="xs:string" minOccurs="0"/>
                        <xs:element name="state" type="xs:string" minOccurs="0" />
                        <xs:element name="status" type="xs:string" minOccurs="0" />
                        <xs:element name="message" type="xs:string" minOccurs="0" />
                        <xs:element name="cost" type="tns:AmountType" minOccurs="0"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*MetadataListDocument*

```
    <xs:element name="MetadataListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:MetadataListType" />
    <xs:complexType name="MetadataListType">
        <xs:sequence>
            <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="metadata" minOccurs="0" maxOccurs="1" type=
↪"tns:MetadataType"/>
                    </xs:sequence>
                    <xs:attribute name="id" type="tns:SiteIdType" />
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*MetadataLockDocument*

```
    <xs:element name="MetadataLockDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:MetadataLockType"/>
    <xs:complexType name="MetadataLockType">
        <xs:sequence>
            <xs:element name="id" type="xs:string"></xs:element>
            <xs:element name="user" type="xs:string"></xs:element>
            <xs:element name="expires" type="xs:dateTime"></xs:element>
            <xs:element name="field" type="xs:string" maxOccurs="unbounded" minOccurs=
↪"0"></xs:element>
        </xs:sequence>
    </xs:complexType>
```

*MetadataLockListDocument*

```
    <xs:element name="MetadataLockListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:MetadataLockListType"/>

    <xs:complexType name="MetadataLockListType">
        <xs:sequence>
            <xs:element name="lock" type="tns:MetadataLockType" maxOccurs="unbounded"␣
↪minOccurs="0"></xs:element>
```

```
        </xs:sequence>
    </xs:complexType>
```

*CollectionListDocument*

```
    <xs:element name="CollectionListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:CollectionListType" />
    <xs:complexType name="CollectionListType">
        <xs:sequence>
            <xs:element name="hits" minOccurs="0" maxOccurs="1" type="xs:integer" />
            <xs:element name="collection" minOccurs="0" maxOccurs="unbounded" type="tns:CollectionType"/>
            <xs:element name="facet" minOccurs="0" maxOccurs="unbounded" type="tns:FacetType"/>
            <xs:element name="suggestion" minOccurs="0" maxOccurs="unbounded" type="tns:SuggestionResultType"/>
            <xs:element name="autocomplete" minOccurs="0" maxOccurs="unbounded" type="tns:AutocompleteResponseType"/>
            <xs:element name="nextCursor" type="xs:string" maxOccurs="1" minOccurs="0"></xs:element>
        </xs:sequence>
    </xs:complexType>
```

*CollectionDocument*

```
    <xs:element name="CollectionDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:CollectionType" />
    <xs:complexType name="CollectionType">
        <xs:sequence>
            <xs:element name="loc" type="xs:anyURI" minOccurs="0"/>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0"/>
            <xs:element name="name" type="xs:string" minOccurs="0"/>
            <xs:element name="content" type="tns:CollectionContentType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="project" type="tns:ProjectType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="sequence" type="tns:SequenceType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="metadata" type="tns:MetadataType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="terse" type="tns:GenericType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="merged-access" type="tns:AccessControlMergedType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="externalId" type="tns:ExternalIdentifierType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="absoluteTime" type="xs:boolean"/>
    </xs:complexType>

    <xs:complexType name="CollectionContentType">
        <xs:sequence>
            <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="uri" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="type" type="tns:CollectionContentEntityTypeType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="reference" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

```
            <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="mode" type="tns:CollectionContentModeType" use="optional"/>
        <xs:attribute name="before" type="xs:string" use="optional"/>
        <xs:attribute name="after" type="xs:string" use="optional"/>
        <xs:attribute name="addItems" type="xs:boolean" use="optional"/>
    </xs:complexType>

    <xs:simpleType name="CollectionContentModeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="add"/>
            <xs:enumeration value="remove"/>
            <xs:enumeration value="move"/>
            <xs:enumeration value="update"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="CollectionContentEntityTypeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="item"/>
            <xs:enumeration value="library"/>
            <xs:enumeration value="collection"/>
        </xs:restriction>
    </xs:simpleType>
```

*UserDocument*

```
    <xs:element name="UserDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:UserType" />
    <xs:complexType name="UserType">
        <xs:sequence>
            <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="loc" type="xs:anyURI" minOccurs="0" />  <!-- output only -->
            <xs:element name="userName" type="xs:string"></xs:element>
            <xs:element name="realName" type="xs:string" minOccurs="0"></xs:element>
            <xs:element name="alias" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="password" type="xs:string" minOccurs="0"></xs:element>
            <xs:element name="salt" type="xs:string" minOccurs="0" maxOccurs="1" />
            <xs:element name="groupList" type="tns:GroupListType" maxOccurs="1" minOccurs="0"></xs:element>
            <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0" maxOccurs="1" />
            <xs:element name="origin" type="xs:string" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
        <xs:attribute name="disabled" type="xs:boolean" use="optional"/>
        <xs:attribute name="accessPreserved" type="xs:boolean" use="optional"/>
        <xs:attribute name="remove" type="xs:boolean" use="optional"/>
    </xs:complexType>
```

*UserListDocument*

```
    <xs:element name="UserListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:UserListType"/>
    <xs:complexType name="UserListType">
```

```
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="user" type="tns:UserType" maxOccurs="unbounded"
↪minOccurs="0"></xs:element>
        </xs:sequence>
    </xs:complexType>
```

*GroupDocument*

```
    <xs:element name="GroupDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:GroupType" />
    <xs:complexType name="GroupType">
        <xs:sequence>
            <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="loc" type="xs:anyURI" minOccurs="0" maxOccurs="1" /> <!-
↪- output only -->
            <xs:element name="groupName" type="xs:string" minOccurs="0" maxOccurs="1"/
↪>
            <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1
↪"/>
            <xs:element name="role" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="childGroupList" type="tns:GroupListType" maxOccurs="1"
↪minOccurs="0"></xs:element>
            <xs:element name="parentGroupList" type="tns:GroupListType" maxOccurs="1"
↪minOccurs="0"></xs:element>
            <xs:element name="userList" type="tns:UserListType" maxOccurs="1"
↪minOccurs="0"></xs:element>
            <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0"
↪maxOccurs="1" />
            <xs:element name="origin" type="xs:string" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
        <xs:attribute name="remove" type="xs:boolean" use="optional"/>
    </xs:complexType>
```

*GroupListDocument*

```
    <xs:element name="GroupListDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:GroupListType" />
    <xs:complexType name="GroupListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="group" type="tns:GroupType" maxOccurs="unbounded"
↪minOccurs="0"></xs:element>
        </xs:sequence>
    </xs:complexType>
```

*AuthenticationTokenDocument*

```
    <xs:element name="AuthenticationTokenDocument" xmlns:tns="http://xml.vidispine.
↪com/schema/vidispine" type="tns:AuthenticationTokenType" />
    <xs:complexType name="AuthenticationTokenType">
        <xs:sequence>
            <xs:element name="token" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="user" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*AccessKeyListDocument*

```
    <xs:element name="AccessKeyListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:AccessKeyListType" />
    <xs:complexType name="AccessKeyListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="key" type="tns:AccessKeyType" maxOccurs="unbounded" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
```

*AccessKeyDocument*

```
    <xs:element name="AccessKeyDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:AccessKeyType" />
    <xs:complexType name="AccessKeyType">
        <xs:sequence>
            <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="secret" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="status" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="ACTIVE"/>
                        <xs:enumeration value="INACTIVE"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="created" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*SimpleMetadataDocument*

```
    <xs:element name="SimpleMetadataDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SimpleMetadataType" />
```

*ConformDocument*

```
    <xs:element name="ConformDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ConformType" />

    <xs:complexType name="ConformType">
        <xs:sequence>
            <xs:element name="timeBase" type="tns:TimeBaseType" minOccurs="0" maxOccurs="1" />
            <xs:element name="timeline" type="tns:ConformTimelineType" minOccurs="1" maxOccurs="1" />
            <xs:element name="overlay" type="tns:ConformOverlayType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="textOverlay" type="tns:TextOverlayType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ConformOverlayType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="1" maxOccurs="1"/>
```

```
            <xs:element name="x" type="xs:int" minOccurs="1" maxOccurs="1"/>
            <xs:element name="y" type="xs:int" minOccurs="1" maxOccurs="1"/>
            <xs:element name="interval" type="tns:TimeIntervalType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ConformTimelineType">
        <xs:sequence>
            <xs:element name="segment" type="tns:ConformSegmentType" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ConformSegmentType">
        <xs:sequence>
            <xs:element name="source" type="tns:ConformSourceType" minOccurs="1" maxOccurs="1" />
            <xs:element name="destination" type="tns:ConformDestinationType" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ConformSourceType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="1" maxOccurs="1" />
            <xs:element name="interval" type="tns:ConformIntervalType" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ConformDestinationType">
        <xs:sequence>
            <xs:element name="interval" type="tns:ConformIntervalType" minOccurs="1" maxOccurs="1" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ConformIntervalType">
        <xs:sequence>
            <xs:element name="start" type="tns:ConformTimePointType" minOccurs="1" maxOccurs="1" />
            <xs:element name="end" type="tns:ConformTimePointType" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ConformTimePointType">
        <xs:sequence>
            <xs:element name="samples" type="xs:integer" minOccurs="1" maxOccurs="1" />
            <xs:element name="timeBase" type="tns:TimeBaseType" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
    </xs:complexType>
```

*JobPoolDocument*

```
    <xs:element name="JobPoolDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:JobPoolType" />
    <xs:complexType name="JobPoolType">
        <xs:sequence>
            <xs:element name="priorityThreshold" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="size" type="xs:int" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*MetricsConfigurationDocument*

```
    <xs:element name="MetricsConfigurationDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:MetricsConfigurationType" />
    <xs:complexType name="MetricsConfigurationType">
        <xs:sequence>
            <xs:element name="statsd" type="tns:StatsdReporterType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="StatsdReporterType">
        <xs:complexContent>
            <xs:extension base="tns:MetricsReporterType">
                <xs:sequence maxOccurs="1" minOccurs="1">
                    <xs:element name="host" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="port" type="xs:int" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="prefix" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="tags" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="MetricsReporterType">
        <xs:sequence>
            <xs:element name="exclude" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="include" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*IndexingConfigurationDocument*

```
    <xs:element name="IndexingConfigurationDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:IndexingConfigurationType" />
    <xs:complexType name="IndexingConfigurationType">
        <xs:sequence>
            <xs:choice>
                <xs:sequence>
                    <xs:element name="solrPath" type="xs:string" minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
```

```
                <xs:sequence>
                    <xs:element name="solrCollection" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    <xs:element name="zookeeperHost" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:sequence>
                    <xs:element name="elasticsearchPath" type="xs:string" minOccurs="1" maxOccurs="1"/>
                </xs:sequence>
            </xs:choice>
            <xs:element name="commitInterval" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="softCommitInterval" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="autoSoftCommit" type="xs:boolean" minOccurs="0" maxOccurs="1"/>

            <xs:element name="pingAttempts" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="pingTimeout" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="queryTimeout" type="xs:int" minOccurs="0" maxOccurs="1"/>

            <xs:element name="fieldDefault" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="fullText" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*BulkyMetadataConfigurationDocument*

```
    <xs:element name="BulkyMetadataConfigurationDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:BulkyMetadataConfigurationType" />
    <xs:complexType name="BulkyMetadataConfigurationType">
        <xs:sequence>
            <xs:element name="uri" type="xs:anyURI" minOccurs="0" maxOccurs="1"/>
            <xs:element name="storageDisabled" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="status" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="metadataInDatabase" type="xs:long"/>
                        <xs:element name="metadataOnStorage" type="xs:long"/>
                        <xs:element name="storageStatus" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*PathAliasConfigurationDocument*

```
    <xs:element name="PathAliasConfigurationDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:PathAliasConfigurationType" />
    <xs:complexType name="PathAliasConfigurationType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="alias" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*JobPoolListDocument*

```
    <xs:element name="JobPoolListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:JobPoolListType" />
    <xs:complexType name="JobPoolListType">
        <xs:sequence>
            <xs:element name="concurrentJobs" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="pool" type="tns:JobPoolType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*FtpPoolConfigurationDocument*

```
    <xs:element name="FtpPoolConfigurationDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:FtpPoolConfigurationType" />
    <xs:complexType name="FtpPoolConfigurationType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="pool" type="tns:ConnectionPoolType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*TaskGroupListDocument*

```
    <xs:element name="TaskGroupListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:TaskGroupListType" />
    <xs:complexType name="TaskGroupListType">
        <xs:sequence>
            <xs:element name="group" type="tns:TaskGroupType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*TaskGroupDocument*

```
    <xs:element name="TaskGroupDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:TaskGroupType" />
    <xs:complexType name="TaskGroupType">
        <xs:sequence>
            <xs:element name="loc" type="xs:anyURI" minOccurs="0" maxOccurs="1"/>
            <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="job" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
```

```
                        <xs:element name="priority" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="user" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="group" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="data" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="key" type="xs:string" minOccurs="1" maxOccurs="1"/>
                                    <xs:element name="value" type="xs:string" minOccurs="0" maxOccurs="1"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="transcoder" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="xs:string" minOccurs="1" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0" maxOccurs="1" />
            <xs:element name="priority" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="maxConcurrency" type="xs:int" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ConnectionPoolType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="minSize" type="xs:int" />
            <xs:element name="maxSize" type="xs:int" minOccurs="0" />
            <xs:element name="evictionInterval" type="xs:int" minOccurs="0" />
            <xs:element name="minIdleTime" type="xs:int" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>
```

*SiteRuleDocument*

```
    <xs:element name="SiteRuleDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SiteRuleType" />
    <xs:complexType name="SiteRuleType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0" />
            <xs:element name="site" type="xs:string" minOccurs="0" />
            <xs:element name="metadata" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="shape" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="groups" type="xs:boolean" minOccurs="0" maxOccurs="1" />
            <xs:element name="access" type="xs:boolean" minOccurs="0" maxOccurs="1" />
```

```
            <xs:element name="files" type="xs:boolean" minOccurs="0" maxOccurs="1" />
            <xs:element name="targetStorage" type="xs:string" minOccurs="0" maxOccurs="1" />
            <xs:element name="localTargetStorage" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="deleted" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*SiteRuleListDocument*

```
    <xs:element name="SiteRuleListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SiteRuleListType" />

    <xs:complexType name="SiteRuleListType">
        <xs:sequence>
            <xs:element name="siteRule" type="tns:SiteRuleType" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
```

*StorageImportDocument*

```
    <xs:element name="StorageImportDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:StorageImportType" />
    <xs:complexType name="StorageImportType">
        <xs:sequence>
            <xs:element name="file" type="tns:FileImportDefType" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FileImportDefType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence>
            <xs:element name="fileId" type="tns:SiteIdType"/>
            <xs:element name="path" type="xs:string" />
            <xs:element name="size" type="xs:long" />
            <xs:element name="component" type="tns:SiteIdType" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
```

*VersionDocument*

```
    <xs:element name="VersionDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:VersionType"/>

    <xs:complexType name="VersionType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="component" type="tns:CompType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="systemInfo" type="tns:SystemInfoType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="licenseInfo" type="tns:LicenseType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
```

```
    <xs:complexType name="SystemInfoType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence maxOccurs="1" minOccurs="0">
            <xs:element name="macaddress" type="xs:string" maxOccurs="unbounded" minOccurs="0"/>
            <xs:element name="databaseSize" type="xs:long" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CompType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="siteId" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="version" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="LicenseType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="expiryDate" type="xs:string" minOccurs="0"/>
            <xs:element name="macaddresses" type="tns:SystemInfoType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="fileStatus" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="storageNumber" type="tns:LicenseNumberType" minOccurs="0"/>
            <xs:element name="userNumber" type="tns:LicenseNumberType" minOccurs="0"/>
            <xs:element name="itemNumber" type="tns:LicenseNumberType" minOccurs="0"/>
            <xs:element name="transcoderNumber" type="tns:LicenseNumberType" minOccurs="0"/>
            <xs:element name="databaseSizeLimit" type="tns:LicenseNumberType" minOccurs="0"/>
            <xs:element name="endCustomerCompanyname" type="xs:string" minOccurs="0"/>
            <xs:element name="endCustomerCompanyContactEmail" type="xs:string" minOccurs="0"/>
            <xs:element name="resellerCompanyName" type="xs:string" minOccurs="0"/>
            <xs:element name="resellerCompanyContactEmail" type="xs:string" minOccurs="0"/>
            <xs:element name="licenseStatus" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="codecStatus" type="tns:CodecStatusType" minOccurs="0"/>
            <xs:element name="licenseErrorStatus" type="tns:LicenseErrorType" minOccurs="0"/>
            <xs:element name="licenseType" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*MasterLicenseDocument*

```
    <xs:element name="MasterLicenseDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:MasterLicenseType" />
    <xs:complexType name="MasterLicenseType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:complexContent>
```

```
            <xs:extension base="tns:LicenseType">
                <xs:sequence maxOccurs="1" minOccurs="1">
                    <xs:element name="masterIdentifier" type="xs:string" minOccurs="1
↪" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

*SlaveLicenseDocument*

```
    <xs:element name="SlaveLicenseDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:SlaveLicenseType" />
    <xs:complexType name="SlaveLicenseType" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine">
        <xs:complexContent>
            <xs:extension base="tns:LicenseType">
                <xs:sequence maxOccurs="1" minOccurs="1">
                    <xs:element name="masterIdentifier" type="xs:string" minOccurs="0
↪" maxOccurs="1"/>
                    <xs:element name="slaveIdentifier" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
                    <xs:element name="slaveInstances" type="xs:integer" minOccurs="0"␣
↪maxOccurs="1"/>
                    <xs:element name="validityTime" type="xs:dateTime" minOccurs="0"␣
↪maxOccurs="1"/>
                    <xs:element name="validityPeriod" type="xs:integer" minOccurs="0"␣
↪maxOccurs="1"/>
                    <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1
↪"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

*SlaveLicenseListDocument*

```
    <xs:element name="SlaveLicenseListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:SlaveLicenseListType" />
    <xs:complexType name="SlaveLicenseListType" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="slaveLicense" type="tns:SlaveLicenseType" maxOccurs=
↪"unbounded" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
```

*SlaveAuthDocument*

```
    <xs:element name="SlaveAuthDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:SlaveAuthType" />
    <xs:complexType name="SlaveAuthType" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="slaveId" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="slaveIp" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="slaveMac" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="requestSourceIp" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
```

```
            <xs:element name="slaveInstanceName" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="hostname" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="licenseStatus" type="tns:VersionType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*SlaveListDocument*

```
    <xs:element name="SlaveListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SlaveListType" />
    <xs:complexType name="SlaveListType">
        <xs:sequence>
            <xs:element name="slave" type="tns:SlaveType" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SlaveType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="Id" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="slaveId" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="slaveIp" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="slaveMac" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="requestSourceIp" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="slaveInstanceName" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="hostname" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="lastUpdated" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*SlaveAuthInfoDocument*

```
    <xs:element name="SlaveAuthInfoDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SlaveAuthInfoType" />
    <xs:complexType name="SlaveAuthInfoType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="masterHost" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
            <xs:element name="slaveId" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

     <xs:complexType name="LicenseNumberType">
        <xs:sequence minOccurs="0" maxOccurs="1">
            <xs:element name="allowed" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="current" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CodecStatusType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
```

```
        <xs:sequence>
            <xs:element name="codec" type="tns:CodecType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="codecExtraTags" type="xs:string" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CodecType">
        <xs:sequence>
            <xs:element name="encode" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="decode" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>

    <xs:complexType name="EncodeDecodeType">
        <xs:sequence maxOccurs="1" minOccurs="0">
            <xs:element name="encode"  type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="decode"  type="xs:boolean" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>


    <xs:complexType name="LicenseErrorType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="licenseError" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <!-- START PROJECT TYPES -->
```

*ProjectFileDocument*

```
    <xs:element name="ProjectFileDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ProjectFileType" />
    <xs:complexType name="ProjectFileType">
        <xs:sequence>
            <xs:element name="location" type="xs:anyURI"/>
            <xs:element name="type" type="xs:string" minOccurs="0"/>
            <xs:element name="asset" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="xs:string"/>
                        <xs:element name="name" type="xs:string"/>
                        <xs:element name="type" type="xs:string"/>
                        <xs:element name="status" type="xs:string" minOccurs="0"/>
                        <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:attribute name="id" type="tns:SiteIdType"/>
                                <xs:attribute name="shape" type="tns:SiteIdType" use="optional"/>
                                <xs:attribute name="match" type="xs:string"/>
                                <xs:attribute name="permission" type="xs:string"/>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="file" type="tns:FileReferenceType" minOccurs="0" maxOccurs="unbounded"/>
```

```
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="FileReferenceType">
        <xs:sequence>
            <!-- Either an id or path will be available, depending on the NLE -->
            <xs:choice>
                <xs:element name="id" type="xs:string"/>
                <xs:element name="path"  type="xs:anyURI"/>
            </xs:choice>
            <xs:element name="hash"  type="xs:string" minOccurs="0"/>
            <xs:element name="status" type="xs:string" minOccurs="0"/>
            <xs:element name="file" type="tns:FileType" minOccurs="0" maxOccurs=
↪"unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*ExportRequestDocument*

```
    <xs:element name="ExportRequestDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:ExportRequestType" />
    <xs:complexType name="ExportRequestType">
        <xs:sequence>
            <xs:element name="tag" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="format" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="content" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
            <xs:element name="storage" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="tns:SiteIdType" minOccurs="1"␣
↪maxOccurs="1"/>
                        <xs:element name="path" type="xs:anyURI" minOccurs="0"␣
↪maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="tns:SiteIdType" minOccurs="1"␣
↪maxOccurs="1"/>
                        <xs:element name="path" type="xs:anyURI" minOccurs="1"␣
↪maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="sequence" type="tns:SequenceType" minOccurs="0"␣
↪maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*ExportResponseDocument*

```
    <xs:element name="ExportResponseDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ExportResponseType" />
    <xs:complexType name="ExportResponseType">
        <xs:sequence>
            <xs:element name="problem" type="tns:ExportProblemType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="mappings" type="tns:EssenceMappingsType" minOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*ExportStatusDocument*

```
    <xs:element name="ExportStatusDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ExportStatusType" />
    <xs:complexType name="ExportStatusType">
        <xs:sequence>
            <xs:element name="problem" type="tns:ExportProblemType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ExportProblemType">
        <xs:sequence>
            <xs:element name="type" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="message" type="xs:string"/>
            <xs:element name="asset" type="xs:string" minOccurs="0"/>
            <xs:element name="parameter" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*EssenceMappingsDocument*

```
    <xs:element name="EssenceMappingsDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:EssenceMappingsType" />
    <xs:complexType name="EssenceMappingsType">
        <xs:sequence>
            <xs:element name="asset" type="tns:AssetMappingType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="file" type="tns:FileMappingType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="storage" type="tns:StorageMappingType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AssetMappingType">
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xs:attribute name="item" type="tns:SiteIdType" use="required"/>
        <xs:attribute name="shape" type="tns:SiteIdType" use="optional"/>
    </xs:complexType>

    <xs:complexType name="StorageMappingType">
        <xs:attribute name="path" type="xs:string" use="required"/>
        <xs:attribute name="id" type="tns:SiteIdType" use="required"/>
    </xs:complexType>

    <xs:complexType name="FileMappingType">
```

```
        <!-- Either an id or path should be provided, depending on the NLE -->
        <xs:attribute name="id" type="xs:string" use="optional"/>
        <xs:attribute name="path"  type="xs:anyURI" use="optional"/>
        <xs:attribute name="hash"  type="xs:string" use="optional"/>
        <xs:attribute name="size" type="xs:long" use="optional"/>
        <xs:attribute name="timestamp" type="xs:dateTime" use="optional"/>
    </xs:complexType>

    <!-- END PROJECT TYPES -->
```

*ReindexRequestDocument*

```
    <xs:element name="ReindexRequestDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ReindexRequestType"/>
    <xs:complexType name="ReindexRequestType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="index" type="xs:string" />
            <xs:element name="priority" type="xs:int" />
            <xs:element name="status" type="xs:string"/>
            <xs:element name="start" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="finish" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="indexesDone" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="indexesTotal" type="xs:integer" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*PlaceholderImportRequestDocument*

```
    <xs:element name="PlaceholderImportRequestDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:PlaceholderImportRequestType"/>
    <xs:complexType name="PlaceholderImportRequestType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="container" type="xs:anyURI" minOccurs="0" maxOccurs="1"/>
            <xs:element name="video" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="audio" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="binary" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
            <!--<xs:element name="metadata" type="tns:MetadataType" minOccurs="0" maxOccurs="1"/>-->
        </xs:sequence>
    </xs:complexType>
```

*ConformRequestDocument*

```
    <xs:element name="ConformRequestDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ConformRequestType"/>
    <xs:complexType name="ConformRequestType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="conform" type="tns:ConformType" minOccurs="1" maxOccurs="1"/>
```

```
            <xs:element name="metadata" type="tns:MetadataType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*SequenceRenderRequestDocument*

```
    <xs:element name="SequenceRenderRequestDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SequenceRenderRequestType"/>
    <xs:complexType name="SequenceRenderRequestType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="sequence" type="tns:SequenceType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="metadata" type="tns:MetadataType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*VidispineServiceListDocument*

```
    <xs:element name="VidispineServiceListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:VidispineServiceListType" />
    <xs:complexType name="VidispineServiceListType">
        <xs:sequence>
            <xs:element name="service" type="tns:VidispineServiceType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*VidispineServiceDocument*

```
    <xs:element name="VidispineServiceDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:VidispineServiceType"/>
    <xs:complexType name="VidispineServiceType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence>
            <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="class" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="arguments" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="isEnabled" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="isRunning" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="exception" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="exceptionTimestamp" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="thread" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="threadStatus" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="load5" type="xs:double" minOccurs="0" maxOccurs="1"/>
            <xs:element name="load60" type="xs:double" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*ExportLocationListDocument*

```
    <xs:element name="ExportLocationListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ExportLocationListType"/>
    <xs:complexType name="ExportLocationListType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence>
            <xs:element name="exportLocation" type="tns:ExportLocationType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*ExportLocationDocument*

```
    <xs:element name="ExportLocationDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ExportLocationType"/>
    <xs:complexType name="ExportLocationType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" />
            <xs:element name="uri" type="xs:string" minOccurs="0" /> <!-- preserved for backwards compatibility -->
            <xs:element name="uriList" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="projection" type="xs:string" minOccurs="0" />
            <xs:element name="tag" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="script" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>


    <!-- SELF TEST DOCUMENTS -->

        <xs:element name="SelfTestDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SelfTestType"/>
        <xs:complexType name="SelfTestType">
                <xs:sequence>
                        <xs:element name="message" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="test" type="tns:SelfTestType" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="name" type="xs:string" use="optional"/>
                <xs:attribute name="description" type="xs:string" use="optional"/>
                <xs:attribute name="status" type="xs:string" use="required"/>
                <xs:attribute name="took" type="xs:string" use="optional"/>
        </xs:complexType>

    <xs:complexType name="LoudnessMixType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" />
            <xs:element name="weightdB" type="xs:double" minOccurs="0" />
            <xs:element name="sourceItemTrack" type="xs:string" minOccurs="0" />
            <xs:element name="sourceStream" type="xs:int" minOccurs="0" />
            <xs:element name="sourceChannel" type="xs:int" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>
```

*LoudnessDocument*

```
    <xs:element name="LoudnessDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:LoudnessType"/>
    <xs:complexType name="LoudnessType" xmlns:tns="http://xml.vidispine.com/schema/vidispine">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0" />
            <xs:element name="shape" type="tns:SiteIdType" minOccurs="0" />
            <xs:element name="shapeTag" type="xs:string" minOccurs="0" />
            <xs:element name="mix" type="tns:LoudnessMixType" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="start" type="xs:string" minOccurs="0" />
            <xs:element name="end" type="xs:string" minOccurs="0" />
            <xs:element name="startLoudness" type="xs:string" minOccurs="0" />
            <xs:element name="endLoudness" type="xs:string" minOccurs="0" />
            <xs:element name="startRange" type="xs:string" minOccurs="0" />
            <xs:element name="endRange" type="xs:string" minOccurs="0" />
            <xs:element name="loudnessLU" type="xs:double" minOccurs="0" />
            <xs:element name="loudnessRangeLU" type="xs:double" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>
```

*AutoProjectionRuleDocument*

```
    <xs:element name="AutoProjectionRuleDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:AutoProjectionRuleType"/>

    <xs:complexType name="AutoProjectionRuleType">
        <xs:sequence>
            <xs:element name="step" type="tns:AutoProjectionStepType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="inputFilters" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="inputFilter" minOccurs="0" maxOccurs="unbounded" type="tns:AutoProjectionInputFilterType"/>
                        <xs:element name="bulkyMetadataKeysRegex" minOccurs="0" maxOccurs="1" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="triggers" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="trigger" minOccurs="0" maxOccurs="unbounded" type="tns:AutoProjectionTriggerType"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="AutoProjectionTriggerType">
```

```
        <xs:restriction base="xs:string">
            <xs:enumeration value="shapeMetadata"/>
            <xs:enumeration value="itemMetadata"/>
            <xs:enumeration value="bulkyMetadata"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="AutoProjectionInputFilterType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="oldMetadata"/>
            <xs:enumeration value="shapeDocument"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="AutoProjectionStepType">
        <xs:sequence>
            <xs:element name="order" type="xs:integer" default="1" minOccurs="0" maxOccurs="1"/>
            <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:sequence>
            <xs:choice>
                <xs:element name="script" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="xslt" type="xs:string" minOccurs="0" maxOccurs="1"/>
            </xs:choice>
        </xs:sequence>
        </xs:sequence>
    </xs:complexType>
```

*MetadataWrapperDocument*

```
    <xs:element name="MetadataWrapperDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:MetadataWrapperType"/>
    <xs:complexType name="MetadataWrapperType">
        <xs:sequence>
            <xs:element name="metadata" type="tns:MetadataType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="oldMetadata" type="tns:MetadataListType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="shape" type="tns:ShapeType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="shapeMetadata" type="tns:SimpleMetadataType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="bulkyMetadata" type="tns:BulkyMetadataType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="oldBulkyMetadata" type="tns:BulkyMetadataType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="targetId" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*ErrorLogListDocument*

```
    <xs:element name="ErrorLogListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ErrorLogListType" />
    <xs:complexType name="ErrorLogListType">
      <xs:sequence>
```

```
        <xs:element name="errorLog" minOccurs="0" maxOccurs="unbounded" type="tns:
↪ErrorLogType"/>
      </xs:sequence>
    </xs:complexType>
```

*ErrorLogDocument*

```
    <xs:element name="ErrorLogDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:ErrorLogType"/>
    <xs:complexType name="ErrorLogType">
        <xs:sequence>
            <xs:element name="id" type="xs:long" minOccurs="0" maxOccurs="1"/>
            <xs:element name="timestamp" type="xs:dateTime" minOccurs="0" maxOccurs="1
↪"/>
            <xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1
↪"/>
            <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="entityType" type="xs:string" minOccurs="0" maxOccurs="1
↪"/>
            <xs:element name="entityId" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*vsXSLTVersion*

```
    <xs:element name="vsXSLTVersion" type="xs:integer"/>
```

*ExportInformationDocument*

```
    <xs:element name="ExportInformationDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:ExportInformationType"/>
    <xs:complexType name="ExportInformationType">
        <xs:sequence>
            <xs:element name="metadataList" type="tns:MetadataListType" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="job" type="tns:JobType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*MetadataSchemaMigrationListDocument*

```
    <xs:element name="MetadataSchemaMigrationListDocument" xmlns:tns="http://xml.
↪vidispine.com/schema/vidispine" type="tns:MetadataSchemaMigrationListType"/>
    <xs:complexType name="MetadataSchemaMigrationListType">
        <xs:sequence>
            <xs:element name="migration" type="tns:MetadataSchemaMigrationType"␣
↪minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*MetadataSchemaMigrationDocument*

```
    <xs:element name="MetadataSchemaMigrationDocument" xmlns:tns="http://xml.
↪vidispine.com/schema/vidispine" type="tns:MetadataSchemaMigrationType"/>
    <xs:complexType name="MetadataSchemaMigrationType">
        <xs:sequence>
            <xs:element name="done" type="xs:boolean" minOccurs="0"/>
            <xs:element name="migrationsLeft" type="xs:integer" minOccurs="0"/>
```

```
            <xs:element name="move" type="tns:MetadataSchemaMoveOperationType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="rename" type="tns:MetadataSchemaRenameOperationType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="delete" type="tns:MetadataSchemaDeleteOperationType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:integer"/>
    </xs:complexType>

    <xs:complexType name="MetadataSchemaMoveOperationType">
        <xs:sequence>
            <xs:element name="from" type="tns:MetadataSchemaHierarchyType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="to" type="tns:MetadataSchemaHierarchyType" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="type" type="xs:string"/>
    </xs:complexType>

    <xs:complexType name="MetadataSchemaDeleteOperationType">
        <xs:sequence>
            <xs:element name="target" type="tns:MetadataSchemaHierarchyType" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="type" type="xs:string"/>
    </xs:complexType>

    <xs:complexType name="MetadataSchemaRenameOperationType">
        <xs:sequence>
            <xs:element name="from" type="tns:MetadataSchemaHierarchyType" minOccurs="1" maxOccurs="1"/>
            <xs:element name="to" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="MetadataSchemaHierarchyType">
        <xs:choice>
            <xs:element name="group" type="tns:MetadataFieldGroupType"/>
            <xs:element name="field" type="tns:MetadataFieldType"/>
        </xs:choice>
    </xs:complexType>
        <xs:element name="EssenceVersionListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:EssenceVersionListType"/>
        <xs:complexType name="EssenceVersionListType">
                <xs:sequence>
                    <xs:element name="version" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                    <xs:sequence>
                                            <xs:element name="id" type="xs:int" minOccurs="1" maxOccurs="1" />
                                            <xs:element name="uri" type="xs:string" minOccurs="1" maxOccurs="1" />
                                            <xs:element name="created" type="xs:dateTime" minOccurs="0" maxOccurs="1" />
                                    </xs:sequence>
                            </xs:complexType>
                    </xs:element>
```

```
                </xs:sequence>
        </xs:complexType>

        <xs:element name="EssenceVersionDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:EssenceVersionType"/>
        <xs:complexType name="EssenceVersionType">
                <xs:sequence>
                        <xs:element name="created" type="xs:dateTime" minOccurs="0" maxOccurs="1" />
                        <xs:element name="shape" type="tns:ShapeType" minOccurs="0" maxOccurs="unbounded" />
                </xs:sequence>
        </xs:complexType>

  <xs:element name="DocumentListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:DocumentListType"/>
  <xs:complexType name="DocumentListType">
    <xs:sequence>
        <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
        <xs:element name="document" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1" />
              <xs:element name="uri" type="xs:string" minOccurs="1" maxOccurs="1" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
    </xs:sequence>
  </xs:complexType>
```

*VXAJobListDocument*

```
    <xs:element name="VXAJobListDocument" type="tns:VXAJobListType"/>
    <xs:complexType name="VXAJobListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer"/>
            <xs:element name="job" type="tns:VXAJobType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*VXAJobDocument*

```
    <xs:element name="VXAJobDocument" type="tns:VXAJobType"/>
    <xs:complexType name="VXAJobType">
        <xs:sequence>
            <xs:element name="vxaId" type="xs:string"/>
            <xs:element name="vxaName" type="xs:string"/>
            <xs:element name="user" type="xs:string"/>
            <xs:element name="jobId" type="xs:long"/>
            <xs:element name="uuid" type="xs:string"/>
            <xs:element name="type" type="xs:string"/>
            <xs:element name="instance" type="xs:string"/>
            <xs:element name="status" type="xs:string"/>
            <xs:element name="errorMessage" type="xs:string"/>
            <xs:element name="progress" type="xs:double"/>
            <xs:element name="itemId" type="xs:string"/>
            <xs:element name="filename" type="xs:string"/>
```

```
            <xs:element name="startTime" type="xs:dateTime"/>
            <xs:element name="plugin" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attribute name="name" type="xs:string"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="jobConfiguration" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="configuration" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*WebsocketUpdateListDocument*

```
    <xs:element name="WebsocketUpdateListDocument" type="tns:WebsocketUpdateListType"/>
    <xs:complexType name="WebsocketUpdateListType">
        <xs:sequence>
            <xs:element name="entry" type="tns:WebsocketUpdateType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*WebsocketUpdateDocument*

```
    <xs:element name="WebsocketUpdateDocument" type="tns:WebsocketUpdateType"/>
    <xs:complexType name="WebsocketUpdateType">
        <xs:choice>
            <xs:element name="item" type="tns:ItemType"/>
            <xs:element name="job" type="tns:JobType"/>
            <xs:element name="vxaJob" type="tns:VXAJobType"/>
            <xs:element name="quota" type="tns:QuotaRuleType"/>
            <xs:element name="storage" type="tns:StorageType"/>
            <xs:element name="vxa" type="tns:VXAUpdateType"/>
        </xs:choice>
    </xs:complexType>
```

*WebsocketUpdateRequestDocument*

```
    <xs:element name="WebsocketUpdateRequestDocument" type="tns:WebsocketUpdateRequestType"/>
    <xs:complexType name="WebsocketUpdateRequestType">
        <xs:sequence>
            <xs:element name="item" type="tns:ItemSearchType" minOccurs="0"/>
            <xs:element name="job" type="tns:JobFilterType" minOccurs="0"/>
            <xs:element name="vxaJob" type="tns:JobFilterType" minOccurs="0"/>
            <xs:element name="quota" type="tns:QuotaFilterType" minOccurs="0"/>
            <xs:element name="storage" type="tns:StorageFilterType" minOccurs="0"/>
            <xs:element name="vxa" type="tns:IDFilterType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="JobFilterType">
```

```
        <xs:sequence>
            <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="username" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="state" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="jobmetadata" type="tns:SimpleMetadataType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="QuotaFilterType">
        <xs:sequence>
            <xs:element name="username" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="groupname" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="exceeded" type="xs:boolean" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="StorageFilterType">
        <xs:sequence>
            <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="vxaId" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="IDFilterType">
        <xs:sequence>
            <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*VXAUpdateDocument*

```
    <xs:element name="VXAUpdateDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:VXAUpdateType" />
    <xs:complexType name="VXAUpdateType">
        <xs:sequence>
            <xs:element name="vxaId" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="user" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="added" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="deleted" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="removedShare" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="addedShare" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="ExportTemplateArchiveType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ZIP"/>
```

```
            <xs:enumeration value="TAR"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ExportTemplateCompressType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="GZ"/>
            <xs:enumeration value="BZIP2"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:group name="ExportTemplateChoiceGroup">
    <xs:choice>
        <xs:element name="archive" type="tns:ExportTemplateArchive" />
        <xs:element name="collection" type="tns:ExportTemplateCollection" />
        <xs:element name="component" type="tns:ExportTemplateComponent" />
        <xs:element name="componentfile" type="tns:ExportTemplateComponentFile" />
        <xs:element name="compress" type="tns:ExportTemplateCompress" />
        <xs:element name="dummy" type="tns:ExportTemplateDummy" />
        <xs:element name="external" type="tns:ExportTemplateExternal" />
        <xs:element name="folder" type="tns:ExportTemplateFolder" />
        <xs:element name="item" type="tns:ExportTemplateItem" />
        <xs:element name="iterate" type="tns:ExportTemplateIterate" />
        <xs:element name="library" type="tns:ExportTemplateLibrary" />
        <xs:element name="sequence" type="tns:ExportTemplateSequence" />
        <xs:element name="shape" type="tns:ExportTemplateShape" />
        <xs:element name="text" type="tns:ExportTemplateText" />
    </xs:choice>
    </xs:group>

    <xs:complexType name="ExportTemplateType">
        <xs:sequence>
            <xs:element name="tag" type="xs:string" minOccurs="0" maxOccurs="unbounded
↪"/>
            <xs:element name="dependency" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
            <xs:element name="filter" type="xs:string" minOccurs="0"/>
            <xs:element name="pre" type="tns:ExportTemplateScript" minOccurs="0"/>
            <xs:element name="preRender" type="tns:ExportTemplateScript" minOccurs="0
↪"/>
            <xs:group ref="tns:ExportTemplateChoiceGroup" minOccurs="0" maxOccurs=
↪"unbounded"/>
        </xs:sequence>
        <xs:attribute name="path" type="xs:string" use="optional"/>
    </xs:complexType>

    <xs:complexType name="ExportTemplateFolder">
        <xs:complexContent>
            <xs:extension base="tns:ExportTemplateType"/>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ExportTemplateArchive">
        <xs:complexContent>
            <xs:extension base="tns:ExportTemplateType">
                <xs:attribute name="method" type="tns:ExportTemplateArchiveType" use=
↪"required"/>
            </xs:extension>
```

```
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ExportTemplateCollection">
        <xs:complexContent>
            <xs:extension base="tns:ExportTemplateType">
                <xs:sequence>
                    <xs:element name="collectionId" type="tns:SiteIdType" minOccurs="0
↪" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ExportTemplateLibrary">
        <xs:complexContent>
            <xs:extension base="tns:ExportTemplateType">
                <xs:sequence>
                    <xs:element name="libraryId" type="tns:SiteIdType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ExportTemplateCompress">
        <xs:complexContent>
            <xs:extension base="tns:ExportTemplateType">
                <xs:attribute name="method" type="tns:ExportTemplateCompressType" use=
↪"required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ExportTemplateItem">
        <xs:complexContent>
            <xs:extension base="tns:ExportTemplateType">
                <xs:sequence>
                    <xs:element name="itemId" type="tns:SiteIdType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ExportTemplateIterate">
        <xs:complexContent>
            <xs:extension base="tns:ExportTemplateType">
                <xs:sequence>
                    <xs:element name="value" type="tns:ExportTemplateScript"␣
↪minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ExportTemplateShape">
        <xs:complexContent>
```

```
            <xs:extension base="tns:ExportTemplateType">
                <xs:sequence>
                    <xs:element name="shapeTag" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="generate" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ExportTemplateComponent">
        <xs:complexContent>
            <xs:extension base="tns:ExportTemplateType">
                <xs:sequence>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ExportTemplateComponentFile">
        <xs:complexContent>
            <xs:extension base="tns:ExportTemplateType">
                <xs:sequence>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ExportTemplateSequence">
        <xs:complexContent>
            <xs:extension base="tns:ExportTemplateType">
                <xs:sequence>
                    <xs:element name="document" type="tns:ExportTemplateTextContent" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="generate" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:simpleType name="ExportTemplateScript">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>

    <xs:complexType name="ExportTemplateTextContent" mixed="true">
        <xs:group ref="tns:ExportTemplateChoiceGroup" minOccurs="0" maxOccurs="unbounded"/>
        <xs:attribute name="scripttags" type="xs:boolean" use="optional"/>
    </xs:complexType>

    <xs:complexType name="ExportTemplateText">
        <xs:complexContent>
            <xs:extension base="tns:ExportTemplateType">
                <xs:sequence>
                    <xs:element name="content" minOccurs="0" maxOccurs="1" type="tns:ExportTemplateTextContent"/>
```

```
                    <xs:element name="xslt" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="script" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    <xs:element name="xml" minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:any minOccurs="1" maxOccurs="1" processContents="lax"/>
                            </xs:sequence>
                            <xs:attribute name="indent" type="xs:int" use="optional"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="charset" type="xs:string" use="optional"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ExportTemplateExternal">
        <xs:complexContent>
            <xs:extension base="tns:ExportTemplateType">
                <xs:sequence>
                    <xs:element name="uri" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ExportTemplateDummy">
        <xs:complexContent>
            <xs:extension base="tns:ExportTemplateType"/>
        </xs:complexContent>
    </xs:complexType>
```

*ExportTemplateDocument*

```
    <xs:element name="ExportTemplateDocument" type="tns:ExportTemplateType"/>
```

*LogReportConfigurationDocument*

```
    <xs:element name="LogReportConfigurationDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:LogReportConfigurationType" />
    <xs:complexType name="LogReportConfigurationType">
        <xs:sequence>
            <xs:element name="path" type="xs:string" minOccurs="0" maxOccurs="1"/> <!-- default: /tmp/LogReport, %TEMP%/LogReport -->
            <xs:element name="expiryTime" type="xs:double" minOccurs="0" maxOccurs="1"/> <!-- default: 7 days -->
            <xs:element name="uploadUri" type="xs:anyURI" minOccurs="0" maxOccurs="1"/>
            <xs:element name="certificate" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="clientKey" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="clientCertificate" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
```

```
        </xs:sequence>
    </xs:complexType>
```

*WaveformDataDocument*

```
    <xs:element name="WaveformDataDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:WaveformDataType" />
    <xs:complexType name="WaveformDataType">
        <xs:sequence>
            <xs:element name="value" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="data" type="xs:double" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="stream" type="xs:int" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="channel" type="xs:int" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="itemTrack" type="xs:string" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="fileId" type="tns:SiteIdType" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="filePath" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*ConstraintValueListDocument*

```
    <xs:element name="ConstraintValueListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ConstraintValueListType" />
    <xs:complexType name="ConstraintValueListType">
        <xs:sequence>
            <xs:element name="value" type="tns:constraintValueType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="constraintValueType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="id" type="xs:string" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
```

*MetadataFieldValidationDocument*

```
    <xs:element name="MetadataFieldValidationDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:MetadataFieldValidationType" />
    <xs:complexType name="MetadataFieldValidationType">
        <xs:sequence>
            <xs:element name="field" type="tns:MetadataFieldValueType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="constraint" minOccurs="0" maxOccurs="1">
```

```
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="field" type="tns:MetadataFieldValueType"␣
↪minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*MetadataFieldValueConstraintListDocument*

```
    <xs:element name="MetadataFieldValueConstraintListDocument" xmlns:tns="http://xml.
↪vidispine.com/schema/vidispine" type="tns:MetadataFieldValueConstraintListType" />
    <xs:complexType name="MetadataFieldValueConstraintListType">
        <xs:sequence>
            <xs:element name="constraint" type="tns:MetadataFieldValueConstraintType"␣
↪minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="MetadataFieldValueConstraintType">
        <xs:sequence>
            <xs:element name="field" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="value" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*MetadataDatasetListDocument*

```
    <xs:element name="MetadataDatasetListDocument" xmlns:tns="http://xml.vidispine.
↪com/schema/vidispine" type="tns:MetadataDatasetListType"/>
    <xs:complexType name="MetadataDatasetListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="dataset" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="name" type="xs:string" minOccurs="1"␣
↪maxOccurs="1" />
                        <xs:element name="uri" type="xs:string" minOccurs="1"␣
↪maxOccurs="1" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*CORSConfigurationDocument*

```
    <xs:element name="CORSConfigurationDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:CORSConfigurationType"/>
    <xs:complexType name="CORSConfigurationType">
        <xs:sequence>
            <xs:element name="entry" type="tns:CORSConfigurationEntry" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
```

```
    </xs:complexType>
    <xs:complexType name="CORSConfigurationEntry">
        <xs:sequence>
            <xs:element name="request" type="tns:CORSConfigurationEntryRequest" minOccurs="1" maxOccurs="unbounded"/>
            <xs:element name="response" type="tns:CORSConfigurationEntryResponse" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="CORSConfigurationEntryRequest">
        <xs:sequence>
            <xs:element name="method" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="origin" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="originRegex" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="pathRegex" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="headerRegex" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="CORSConfigurationEntryResponse">
        <xs:sequence>
            <xs:element name="allowOrigin" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="allowMethods" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="allowHeaders" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="allowMaxAge" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="allowOtherHeader" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*DatabasePurgingConfigurationDocument*

```
    <xs:element name="DatabasePurgingConfigurationDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:DatabasePurgingConfigurationType" />
    <xs:complexType name="DatabasePurgingConfigurationType">
        <xs:sequence>
            <xs:element name="changeLog" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="age" minOccurs="0" maxOccurs="1" type="xs:int"/> <!-- remove processed entries older than this number of minutes -->
                        <xs:element name="forceAge" minOccurs="0" maxOccurs="1" type="xs:int"/> <!-- remove any entries older than this number of minutes -->
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="auditTrail" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="age" minOccurs="0" maxOccurs="1" type="xs:int"/> <!-- remove entries older than this number of minutes -->
```

```
                        <xs:element name="uri" minOccurs="0" maxOccurs="1" type="xs:anyURI"/> <!-- store to this URI -->
                        <xs:element name="compress" minOccurs="0" maxOccurs="1" type="xs:boolean"/> <!-- store in compressed form, default true -->
                        <xs:element name="batch" minOccurs="0" maxOccurs="1" type="xs:int"/> <!-- process this number of entries each time, default 10000 (if this many number of entries do not exist, purging will pause -->
                        <xs:element name="body" minOccurs="0" maxOccurs="1" type="xs:boolean"/> <!-- include body, default false -->
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="job" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="age" minOccurs="0" maxOccurs="1" type="xs:int"/> <!-- remove entries older than this number of minutes -->
                        <xs:element name="uri" minOccurs="0" maxOccurs="1" type="xs:anyURI"/> <!-- store to this URI -->
                        <xs:element name="compress" minOccurs="0" maxOccurs="1" type="xs:boolean"/> <!-- store in compressed form, default true -->
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="transferLog" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="age" minOccurs="0" maxOccurs="1" type="xs:int"/> <!-- remove finished entries older than this number of minutes -->
                        <xs:element name="forceAge" minOccurs="0" maxOccurs="1" type="xs:int"/> <!-- remove any entries older than this number of minutes, default same as above -->
                        <xs:element name="uri" minOccurs="0" maxOccurs="1" type="xs:anyURI"/> <!-- store to this URI -->
                        <xs:element name="compress" minOccurs="0" maxOccurs="1" type="xs:boolean"/> <!-- store in compressed form, default true -->
                        <xs:element name="batch" minOccurs="0" maxOccurs="1" type="xs:int"/> <!-- process this number of entries each time, default 10000 (if this many number of entries do not exist, purging will pause -->
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*DeletionLockDocument*

```
    <xs:element name="DeletionLockDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:DeletionLockType" />
    <xs:complexType name="DeletionLockType">
        <xs:sequence>
            <xs:element name="id" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="user" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="expiryTime" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
            <xs:element name="modified" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
            <xs:element name="entityType" type="xs:string" minOccurs="1" maxOccurs="1"/>
```

```
            <xs:element name="entityId" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0"␣
↪maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="isEffective" type="xs:boolean" default="false"/>
        <xs:attribute name="isInherited" type="xs:boolean" default="false"/>
        <xs:attribute name="isExpired" type="xs:boolean" default="false"/>
    </xs:complexType>
```

*DeletionLockListDocument*

```
    <xs:element name="DeletionLockListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:DeletionLockListType" />
    <xs:complexType name="DeletionLockListType">
        <xs:sequence>
            <xs:element name="lock" type="tns:DeletionLockType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*OAuth2ConfigurationDocument*

```
    <xs:element name="OAuth2ConfigurationDocument" xmlns:tns="http://xml.vidispine.
↪com/schema/vidispine" type="tns:OAuth2ConfigurationType" />
    <xs:complexType name="OAuth2ConfigurationType">
        <xs:sequence>
            <xs:element name="federationMetadataURI" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="federationMetadataInterval" type="xs:int" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="expectedAudience" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="validationEndpoint" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="tokenUser" type="xs:string" minOccurs="0" maxOccurs="1"/
↪>
            <xs:element name="x509Certificate" type="xs:string" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="publicKey" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*JobPriorityConfigurationDocument*

```
    <xs:element name="JobPriorityConfigurationDocument" xmlns:tns="http://xml.
↪vidispine.com/schema/vidispine" type="tns:JobPriorityConfigurationType"/>
    <xs:complexType name="JobPriorityConfigurationType">
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="job">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attribute name="type" type="xs:string" use="required"/
↪>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
```

```
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*ThumbnailSpriteSheetDocument*

```
    <xs:element name="ThumbnailSpriteSheetDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ThumbnailSpriteSheetType"/>
    <xs:complexType name="ThumbnailSpriteSheetType">
        <xs:sequence>
            <xs:element name="etag" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="url" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="path" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="thumbnail" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="width" type="xs:int"/>
                        <xs:element name="height" type="xs:int"/>
                        <xs:element name="x" type="xs:int"/>
                        <xs:element name="y" type="xs:int"/>
                        <xs:element name="timecode" type="tns:TimeCodeType"/>
                        <xs:element name="endTimecode" type="tns:TimeCodeType" minOccurs="0" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*AnalyzePresetDocument*

```
    <xs:element name="AnalyzePresetDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:AnalyzePresetType" />

    <xs:complexType name="AnalyzePresetType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="vidispine" type="tns:AnalyzeJobType" />
            <xs:element name="data" type="tns:KeyValueType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*AnalyzePresetListDocument*

```
    <xs:element name="AnalyzePresetListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:AnalyzePresetListType" />
    <xs:complexType name="AnalyzePresetListType">
        <xs:sequence>
            <xs:element name="preset" type="tns:AnalyzePresetType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

```

*ServiceConfigurationResultDocument*

```
    <xs:element name="ServiceConfigurationResultDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ServiceConfigurationResultType"/>
    <xs:complexType name="ServiceConfigurationResultType">
        <xs:sequence>
            <xs:element name="serviceName" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="serviceId" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="configurationVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="preCheck" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
            <xs:element name="success" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
            <xs:element name="message" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="executedSteps" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="order" type="xs:int"/>
                        <xs:element name="description" type="xs:string"/>
                        <xs:element name="resource" type="xs:string"/>
                        <xs:element name="result" type="xs:string"/>
                        <xs:element name="success" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="data" type="xs:string" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

### 17.38.2 common.xsd

Common elements to API and Transcoder.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.vidispine.com/schema/vidispine"
    elementFormDefault="qualified"
    xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
    jaxb:version="1.0"
    xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
    jaxb:extensionBindingPrefixes="xjc" xmlns:tns="http://xml.vidispine.com/schema/vidispine">

    <xs:simpleType name="SiteIdType">
        <xs:restriction base="xs:string">
            <xs:pattern value="([_A-Za-z]+-)?([A-Za-z_][A-Za-z0-9_]*-|(([A-Za-z_][A-Za-z0-9_]*)?\*))([0-9]{1}[0-9]{0,31})"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="UUIDType">
        <xs:restriction base="xs:string">
            <xs:pattern value="[A-Fa-f0-9]{8}-[A-Fa-f0-9]{4}-[A-Fa-f0-9]{4}-[A-Fa-f0-9]{4}-[A-Fa-f0-9]{12}"/>
```

```
        </xs:restriction>
    </xs:simpleType>
```

*URIListDocument*

```
    <xs:element name="URIListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:URIListType" />
    <xs:complexType name="URIListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:integer" minOccurs="0" maxOccurs="1"/>
            <xs:element name="uri" type="xs:anyURI" maxOccurs="unbounded" minOccurs="0"></xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="MetadataSchemaElementType">
        <xs:attributeGroup ref="tns:MetadataSchemaAttributes"/>
        <xs:attribute name="reference" type="xs:boolean" use="optional"/>
    </xs:complexType>

    <xs:attributeGroup name="MetadataSchemaAttributes">
        <!-- Minimum number of elements -->
        <xs:attribute name="min" type="xs:int" use="required"/>
        <!-- Maximum number of elements. A negative number is regarded as infinity. -->
        <xs:attribute name="max" type="xs:int" use="required"/>
        <xs:attribute name="name" type="xs:string" use="optional"/>
    </xs:attributeGroup>
```

*FileDocument*

```
    <xs:element name="FileDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:FileType" />
    <xs:complexType name="FileType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0"/>
            <xs:element name="path" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="uri"  type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="state" type="xs:string"/>
            <xs:element name="size" type="xs:long" minOccurs="0"/>
            <xs:element name="hash" type="xs:string" minOccurs="0"/>
            <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>
            <xs:element name="refreshFlag" type="xs:int" minOccurs="0"/>
            <xs:element name="sequence" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="storage" type="tns:SiteIdType" minOccurs="0"/>
            <xs:element name="storageDefinition" type="tns:StorageType" minOccurs="0"/>
            <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="shape" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="id" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
```

```
                                    <xs:element name="component" minOccurs="0" maxOccurs="unbounded">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element name="id" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0" maxOccurs="1" />
            <xs:element name="range" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="start" type="xs:string" use="required"/>
                    <xs:attribute name="count" type="xs:long" use="required"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Stuff for Shape starts here -->
    <xs:complexType name="ResolutionType">
        <xs:sequence>
            <xs:element name="width" type="xs:unsignedInt"/>
            <xs:element name="height" type="xs:unsignedInt"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="RationalType">
        <xs:sequence>
            <xs:element name="numerator" type="xs:int"/>
            <xs:element name="denominator" type="xs:int"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="TimeBaseType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:RationalType"/>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="FrameRateType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:RationalType"/>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="TimeCodeType">
        <xs:sequence>
            <xs:element name="samples" type="xs:long"/>
            <xs:element name="timeBase" type="tns:TimeBaseType"/>
        </xs:sequence>
    </xs:complexType>
```

```
    <xs:complexType name="TimeIntervalType">
        <xs:sequence>
            <xs:element name="start" type="tns:TimeCodeType" minOccurs="0"/>
            <xs:element name="end"   type="tns:TimeCodeType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="AspectRatioType">
        <xs:sequence>
            <xs:element name="horizontal" type="xs:unsignedInt"/>
            <xs:element name="vertical" type="xs:unsignedInt"/>
        </xs:sequence>
    </xs:complexType>
```

*ComponentListDocument*

```
    <xs:element name="ComponentListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ComponentListType" />

    <xs:complexType name = "ComponentListType">
        <xs:sequence>
            <xs:element name="component" minOccurs="0" maxOccurs="unbounded" type="tns:ComponentType" />
        </xs:sequence>
    </xs:complexType>
```

*ComponentDocument*

```
    <xs:element name="ComponentDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ComponentType" />
    <xs:complexType name="ComponentType">
        <xs:sequence>
            <xs:element name="file" type="tns:FileType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0"/>

            <!-- Flat metadata, meaning a simple list of key-value pairs.
                 This has been put in ComponentType since both ContainerComponent and the MediaComponents
                 have metadata. In other words, files can have both global and stream-specific metadata.
            -->
            <xs:element name="metadata" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*BinaryComponentDocument*

```
    <xs:element name="BinaryComponentDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:BinaryComponentType" />
    <xs:complexType name="BinaryComponentType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:ComponentType">
                <xs:sequence>
                    <xs:element name="format" type="xs:string" minOccurs="0"/>      <!-- Example: idv3 -->
                    <xs:element name="encoding" type="xs:string" minOccurs="0"/>    <!-- Example: base64, gzip -->
```

```
                    <xs:element name="offset" type="xs:long" minOccurs="0"/>
                    <xs:element name="length" type="xs:long" minOccurs="0"/>
                    <xs:element name="mediaInfo" type="tns:BaseMediaInfoType"␣
↪minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

*ContainerComponentDocument*

```
    <xs:element name="ContainerComponentDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:ContainerComponentType" />
    <xs:complexType name="ContainerComponentType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
↪"tns:ComponentType">
                <xs:sequence>
                    <xs:element name="duration" type="tns:TimeCodeType" minOccurs="0"/
↪>
                    <xs:element name="format" type="xs:string" minOccurs="0"/>

                    <!-- Sub format.
                         This field was created for transcoder ticket #186.
                         Its exact function isn't set in stone yet.
                         Possible values:

                         * Unset
                         * "mxf_d10"
                    -->
                    <xs:element name="subFormat" type="xs:string" minOccurs="0"/>

                    <xs:element name="firstSMPTETimecode" type="xs:string" minOccurs=
↪"0"/>

                    <!-- Corresponds to StartTimecode in TimecodeComponent in MXF -->
                    <xs:element name="startTimecode" type="xs:long" minOccurs="0"/>

                    <!-- First timestamp in container -->
                    <xs:element name="startTimestamp" type="tns:TimeCodeType"␣
↪minOccurs="0"/>

                    <!-- Corresponds to RoundedTimeBase in TimecodeComponent in MXF --
↪>
                    <xs:element name="roundedTimeBase" type="xs:int" minOccurs="0"/>

                    <!-- Corresponds to DropFrame in TimecodeComponent in MXF -->
                    <xs:element name="dropFrame" type="xs:boolean" minOccurs="0"/>
                    <xs:element name="timeCodeTimeBase" type="tns:TimeBaseType"␣
↪minOccurs="0" />
                    <xs:element name="mediaInfo" type="tns:BaseMediaInfoType"␣
↪minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

*DescriptorComponentDocument*

```
    <xs:element name="DescriptorComponentDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:DescriptorComponentType" />
    <xs:complexType name="DescriptorComponentType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:ComponentType">
                <xs:sequence>
                    <xs:element name="description" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:simpleContent>
                                <xs:extension base="xs:string">
                                    <xs:attribute name="type" type="xs:string" use="required"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

*SubtitleComponentDocument*

```
    <xs:element name="SubtitleComponentDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SubtitleComponentType" />
    <xs:complexType name="SubtitleComponentType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:MediaComponentType">
                <!--<xs:sequence>
                    <xs:element name="description" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:simpleContent>
                                <xs:extension base="xs:string">
                                    <xs:attribute name="type" type="xs:string" use="required"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>-->
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>


    <xs:complexType name="EDLType">
        <xs:sequence>
            <!-- MOV specifies length in terms of the mvhd time scale, which is simply an integer.
                 In other words, NTSC files have a time scale of 2997, not 30000/1001.
                 The transcoder fixes up timestamps given from mov.c, so 1700@2997 becomes 17@NTSC and stores the corrected time scale here.
            -->
            <xs:element name="timeScale" type="tns:TimeBaseType"/>
```

```
            <!-- Explicit time base for EDLEntryType.start values.
                 The purpose of this field is to reduce confusion where exactly the time base for the start values is taken from.
                 Using this field also removes the need for running a shape deduction step in order to figure out the time base for each stream.
                 In other words: not using this field results in hairy behavior.
                 See T#208.
            -->
            <xs:element name="timeBase" type="tns:TimeBaseType" minOccurs="0"/>
            <xs:element name="entry" type="tns:EDLEntryType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="EDLEntryType">
        <xs:attribute name="start" type="xs:long" use="required"/>      <!-- In MediaComponentType/timeBase units -->
        <xs:attribute name="length" type="xs:long" use="required"/>     <!-- In EDLType/timeScale units -->
        <xs:attribute name="mediaRate" type="xs:int" use="required"/>   <!-- 16.16 fixed-point, meaning divide by 65536 to get the actual rate. Non-float to promote losslessness. Should be > 0 -->
    </xs:complexType>

    <xs:complexType name="MediaComponentType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:ComponentType">
                <xs:sequence>
                    <xs:element name="codec" type="xs:string" minOccurs="0"/>
                    <xs:element name="timeBase" type="tns:TimeBaseType" minOccurs="0"/>
                    <xs:element name="itemTrack" type="xs:string" minOccurs="0"/>
                    <xs:element name="essenceStreamId" type="xs:unsignedShort" minOccurs="0"/>
                    <xs:element name="interval" type="tns:TimeIntervalType" minOccurs="0"/>
                    <xs:element name="bitrate" type="xs:unsignedInt" minOccurs="0"/>
                    <xs:element name="numberOfPackets" type="xs:long" minOccurs="0"/>
                    <xs:element name="extradata" type="xs:hexBinary" minOccurs="0"/>
                    <xs:element name="pid" type="xs:int" minOccurs="0"/>
                    <xs:element name="duration" type="tns:TimeCodeType" minOccurs="0"/>          <!-- Length of stream - may be different among components in the same file -->
                    <xs:element name="profile" type="xs:int" minOccurs="0"/>                     <!-- Corresponds to AVCodecContext::profile -->
                    <xs:element name="level" type="xs:int" minOccurs="0"/>                       <!-- Corresponds to AVCodecContext::level -->
                    <xs:element name="edl" type="tns:EDLType" minOccurs="0"/>                    <!-- Edit Decision List - derived from elst tags -->
                    <xs:element name="startTimestamp" type="tns:TimeCodeType" minOccurs="0"/>    <!-- First timestamp in media -->
                    <xs:element name="repeatCount" type="xs:unsignedLong" minOccurs="0"/>
                    <xs:element name="trackOrder" type="xs:int" minOccurs="0"/>
                    <xs:element name="segment" type="xs:int" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
```

```
        </xs:complexContent>
    </xs:complexType>

    <!-- Returns various fields parsed by libmediainfo -->
    <xs:complexType name="BaseMediaInfoType">
        <xs:sequence>
            <xs:element name="Bit_rate_mode" type="xs:string" minOccurs="0"/>
      <!-- "Bit rate mode" like "CBR" or "VBR" -->
            <xs:element name="Source" type="xs:string" minOccurs="0"/>
      <!-- "Source". QT-ref source media file, like "media.dir/foo.m2v" -->
            <xs:element name="property" type="tns:KeyValuePairType" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AudioMediaInfoType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
"tns:BaseMediaInfoType">
                <xs:sequence/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="VideoMediaInfoType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
"tns:BaseMediaInfoType">
                <xs:sequence>
                    <xs:element name="Format_Settings_GOP" type="xs:string" minOccurs=
"0"/>         <!-- "Format settings, GOP0" like "N=1" (intra-only) or "M=2, N=60" --
>
                    <xs:element name="intra_dc_precision" type="xs:int" minOccurs="0"/
>             <!-- "intra_dc_precision". 8, 9, 10 or 11 if set -->
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

*AudioComponentDocument*

```
    <xs:element name="AudioComponentDocument" xmlns:tns="http://xml.vidispine.com/
schema/vidispine" type="tns:AudioComponentType" />
    <xs:complexType name="AudioComponentType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
"tns:MediaComponentType">
                <xs:sequence>
                    <xs:element name="channelCount" type="xs:unsignedShort"/>
                    <xs:element name="channelLayout" type="xs:long" minOccurs="0"/>
                    <xs:element name="sampleFormat" type="xs:string" minOccurs="0"/>
                    <xs:element name="frameSize" type="xs:unsignedInt" minOccurs="0"/>
                    <xs:element name="blockAlign" type="xs:unsignedInt" minOccurs="0"/
>

                    <xs:element name="mediaInfo" type="tns:AudioMediaInfoType"
minOccurs="0"/>
                </xs:sequence>
```

```
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

*VideoComponentDocument*

```
    <xs:element name="VideoComponentDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:VideoComponentType" />
    <xs:complexType name="VideoComponentType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:MediaComponentType">
                <xs:sequence>
                    <xs:element name="videoStandard" minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                            <xs:simpleContent>
                                <xs:extension base="xs:string">
                                    <xs:attribute name="type" type="xs:string" use="required"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="resolution" type="tns:ResolutionType" minOccurs="0"/>
                    <xs:element name="pixelFormat" type="xs:string" minOccurs="0"/>
                    <xs:element name="maxBFrames" type="xs:unsignedShort" minOccurs="0"/>
                    <xs:element name="pixelAspectRatio" type="tns:AspectRatioType" minOccurs="0"/>

                    <!-- Field order
                            "progressive" for progressive video
                            "F1" for interlaced in "top field first" order
                            "F2" for interlaced in "bottom field first" order

                         Note that the meaning of F1 and F2 are opposite to S314m where Field 1 is the bottom (odd) field and vice versa.
                         This was not known when this element was introduced, and changing this is likely to result in problems.
                    -->
                    <xs:element name="fieldOrder" type="xs:string" minOccurs="0"/>
                    <xs:element name="codecTimeBase" type="tns:TimeBaseType" minOccurs="0"/>
                    <xs:element name="averageFrameRate" type="tns:TimeBaseType" minOccurs="0"/>
                    <xs:element name="realBaseFrameRate" type="tns:TimeBaseType" minOccurs="0"/>

                    <!-- MXF-ish display rectangle.
                         This means values straight from CDCIDescriptor for MXF,
                         but scaled down by SAR for clap in MOV.
                         In other words, "to display" space, not "displayed" (aka raster) space.
                    -->
                    <xs:element name="displayWidth" type="tns:RationalType" minOccurs="0"/>
                    <xs:element name="displayHeight" type="tns:RationalType" minOccurs="0"/>
```

```
                    <xs:element name="displayXOffset" type="tns:RationalType" minOccurs="0"/>
                    <xs:element name="displayYOffset" type="tns:RationalType" minOccurs="0"/>

                    <!-- DAR = displayWidth/displayHeight * containerSAR -->
                    <xs:element name="containerSAR" type="tns:AspectRatioType" minOccurs="0"/>

                    <xs:element name="colr_primaries" type="xs:int" minOccurs="0"/>
                    <xs:element name="colr_transfer_function" type="xs:int" minOccurs="0"/>
                    <xs:element name="colr_matrix" type="xs:int" minOccurs="0"/>
                    <xs:element name="max_packet_size" type="xs:int" minOccurs="0"/>

                    <!-- Codec-level time code information - typically set from the 25-bit value in the first MPEG-2 GOP header.
                         Use averageFrameRate in lieu of RoundedTimeBase.
                    -->
                    <xs:element name="startTimecode" type="xs:long" minOccurs="0"/>
                    <xs:element name="dropFrame" type="xs:boolean" minOccurs="0"/>

                    <!-- needed to get H.264 decoding working properly in some cases -->
                    <xs:element name="ticks_per_frame" type="xs:int" minOccurs="0"/>

                    <!-- Added for T#135 -->
                    <!-- Bit depth (default 8) -->
                    <xs:element name="bitDepth" type="xs:int" minOccurs="0"/>
                    <xs:element name="bitsPerPixel" type="xs:int" minOccurs="0"/>

                    <!-- Color primaries in string form (c.f.  colr_primaries above -->
                    <xs:element name="colorPrimaries" type="xs:string" minOccurs="0"/>

                    <xs:element name="mediaInfo" type="tns:VideoMediaInfoType" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

*ShapeDocument*

```
    <xs:element name="ShapeDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:ShapeType"/>
    <xs:complexType name="ShapeType">
        <xs:sequence>
            <xs:element name="id" type="tns:SiteIdType" minOccurs="0"/>
            <xs:element name="created" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>
            <xs:element name="essenceVersion" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="tag" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="mimeType" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="uuid" type="tns:UUIDType" minOccurs="0" maxOccurs="1"/>
```

```
            <xs:element name="binaryComponent" type="tns:BinaryComponentType" minOccurs="0" maxOccurs="unbounded"/>
            <!-- container is optional since we might create a ShapeType which merely helps point to source material -->
            <xs:element name="containerComponent" type="tns:ContainerComponentType" minOccurs="0"/>
            <xs:element name="descriptorComponent" type="tns:DescriptorComponentType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="audioComponent" type="tns:AudioComponentType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="videoComponent" type="tns:VideoComponentType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="subtitleComponent" type="tns:SubtitleComponentType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="metadata" minOccurs="0" maxOccurs="1" type="tns:SimpleMetadataType"/>
            <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="tns:SiteIdType" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <!-- Types for bulky metadata -->
```

*BulkyMetadataDocument*

```
    <xs:element name="BulkyMetadataDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:BulkyMetadataType"/>
    <xs:complexType name="BulkyMetadataType">
        <xs:sequence>
            <xs:element name="field" type="tns:BulkyMetadataPairType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="tns:SiteIdType" use="optional"/>
    </xs:complexType>
    <xs:complexType name="BulkyMetadataPairType">
        <xs:sequence>
            <xs:element name="key" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:choice>
                <xs:element name="value" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="maps" type="tns:BulkyMapListType" minOccurs="1" maxOccurs="1"/>
            </xs:choice>
        </xs:sequence>
        <xs:attribute name="start" type="xs:string" use="optional"/>
        <xs:attribute name="end" type="xs:string" use="optional"/>
        <xs:attribute name="stream" type="xs:int" use="optional"/>
        <xs:attribute name="channel" type="xs:int" use="optional"/>
        <xs:attribute name="itemTrack" type="xs:string" use="optional"/>
    </xs:complexType>
```

*BulkyMapListDocument*

```
    <xs:element name="BulkyMapListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:BulkyMapListType"/>
    <xs:complexType name="BulkyMapListType">
        <xs:sequence>
            <xs:element name="map" type="tns:BulkyMapType" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="BulkyMapType">
        <xs:sequence>
            <xs:element name="entry" type="tns:BulkyMapEntryType" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="BulkyMapEntryType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="key" type="xs:string" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>



    <!-- Types for requesting merging according to a timeline (aka pasting together resources) -->
```

*TimelineJobRequestDocument*

```
    <xs:element name="TimelineJobRequestDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:TimelineJobRequestType" />
    <xs:complexType name="TimelineJobRequestType">
        <xs:complexContent>
            <xs:extension base="tns:TranscoderJobType">
        <xs:sequence>
            <xs:element name="outputUri" type="xs:anyURI"/>
            <xs:element name="containerFormat" type="xs:string"/>
            <xs:element name="stream" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="input" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="uri" type="xs:anyURI"/>
                                    <xs:element name="stream" type="xs:unsignedShort"/>
                                    <xs:element name="interval" type="tns:TimeIntervalType"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>

            </xs:element>

            <!--xs:choice>
                <xs:element name="simpleTimeline" type="tns:SimpleTimelineType"/>
```

```
                <xs:element name="timeline" type="tns:TimelineType"/>
            </xs:choice>
            <xs:element name="thumbnailResourceUri" type="xs:anyURI" minOccurs="0"␣
↪maxOccurs="unbounded"/-->
        </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="ComplexJobOTIFType">
        <xs:sequence>
            <xs:element name="trackerPlugin" type="xs:string" minOccurs="1"/>
            <xs:element name="versionMajor" type="xs:int" minOccurs="1"/>
            <xs:element name="versionMinor" type="xs:int" minOccurs="1"/>
            <xs:element name="versionPatch" type="xs:int" minOccurs="1"/>
            <xs:element name="bulkyMetadataURI" type="xs:anyURI" minOccurs="0"␣
↪maxOccurs="1" />
            <xs:element name="configuration" type="tns:KeyValuePairType" minOccurs="0
↪" maxOccurs="unbounded"/>
            <xs:element name="resource" type="tns:NameURIPairType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Types for requesting a more complex "map this to that" type of transcode job␣
↪-->
    <xs:complexType name="ComplexJobOutputType">
        <xs:sequence>
            <!-- Use multiple id element to multiplex the encoded data to multiple
                 files without having to encode more than once
            -->
            <xs:element name="id" type="xs:int" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="start" type="tns:TimeCodeType" minOccurs="0"/>
            <xs:element name="codec" type="xs:string" minOccurs="0"/>

            <!-- FOURCC. Corresponds to codec_tag in libav* terms,
                 hence the name. Typically a four-character ASCII string.
                 May need to be fairly arbitrary constants though, so
                 an xs:string is insufficient. Hence an int is used.

                 An earlier way of setting the codecTag for four-character
                 strings was to set the first character as the LSB and so on.
                 In other words:
                  "ai55" -> 'a' + ('i' << 8) + ('5' << 16) + ('5' << 24)
                 However, this use has been deprecated.

                 Instead, just set the codecTagString to the character string, as in
                 <codecTagString>ai55</codecTagString>
            -->
            <xs:element name="codecTag" type="xs:unsignedInt" minOccurs="0"/>
            <xs:element name="codecTagString" type="xs:string" minOccurs="0"/>

            <!-- Name that the muxer should use in the output file for the codec.
                 Corresponds to codec_name in libav*.
            -->
            <xs:element name="codecName" type="xs:string" minOccurs="0"/>

            <xs:element name="bitrate" type="xs:unsignedInt" minOccurs="0"/>
```

```
561             <xs:element name="timeBase" type="tns:TimeBaseType" minOccurs="0"/>
562
563             <!-- Profile/presets to use.
564                  MainConcept examples: "ipod", "baseline", "main", "high".
565                  For libavcodec, see presets/ directory. Examples: "main", "normal",
↪"hq".
566             -->
567             <xs:element name="preset" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
568
569             <!-- Edit Decision List -->
570             <xs:element name="edl" type="tns:EDLType" minOccurs="0"/>
571
572             <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
573
574             <xs:element name="objectTracking" type="tns:ComplexJobOTIFType" minOccurs=
↪"0" maxOccurs="1"/>
575
576             <xs:element name="metadata" type="tns:KeyValuePairType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
577         </xs:sequence>
578     </xs:complexType>
579
580     <xs:complexType name="ComplexJobAudioOutputType">
581         <xs:complexContent>
582             <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
↪"tns:ComplexJobOutputType">
583                 <xs:sequence>
584                     <!-- See generateThumbnails/uri -->
585                     <xs:element name="thumbnailUri" type="xs:anyURI" minOccurs="0"␣
↪maxOccurs="unbounded"/>    <!-- If set, upload thumbnail to specified URIs -->
586                     <xs:choice minOccurs="0" maxOccurs="1" >
587                         <xs:element name="channelLayout" type="xs:long"/>           <!
↪-- binary representation, ex: 5.1 => 1551, see ffmpeg channel_layout.h -->
588
589                         <!--* name can be one or several of the following notations,
590                             * separated by '+' or '|':
591                             * - the name of an usual channel layout (mono, stereo, 4.
↪0, quad, 5.0,
592                             *   5.0(side), 5.1, 5.1(side), 7.1, 7.1(wide), downmix);
593                             * - the name of a single channel (FL, FR, FC, LFE, BL, BR,
↪ FLC, FRC, BC,
594                             *   SL, SR, TC, TFL, TFC, TFR, TBL, TBC, TBR, DL, DR);
595                             * - a number of channels, in decimal, followed by 'c',␣
↪yielding
596                             *   the default channel layout for that number of␣
↪channels (@see
597                             *   av_get_default_channel_layout);
598                             * - a channel layout mask, in hexadecimal starting with
↪"0x" (see the
599                             *   AV_CH_* macros).
600                             *
601                             * Example: "stereo+FC" = "2c+FC" = "2c+1c" = "0x7" git -->
602                         <xs:element name="channelLayoutName" type="xs:string"/>
603                     </xs:choice>
604                 </xs:sequence>
605             </xs:extension>
```

```
606         </xs:complexContent>
607     </xs:complexType>
608 
609     <xs:complexType name="OverlayType">
610         <xs:sequence>
611             <!-- URI for image to overlay
612                  Should use a pixel format suitable for alpha blending,
613                  like 8-bit RGBA.
614             -->
615             <xs:element name="uri" type="xs:anyURI" maxOccurs="unbounded"/>
616             <xs:element name="range" type="tns:SequenceRangeType" maxOccurs="unbounded
↪"/>
617             <xs:element name="id" minOccurs="0" maxOccurs="1" type="tns:SiteIdType"/>
618 
619             <!-- Coordinates in video to place overlay at. Can be negative -->
620             <xs:element name="x" type="xs:int"/>
621             <xs:element name="y" type="xs:int"/>
622 
623             <!-- Optional: time interval to perform overlay in -->
624             <xs:element name="interval" type="tns:TimeIntervalType" minOccurs="0"/>
625             <xs:element name="opacity" minOccurs="0" maxOccurs="1" type="xs:int"/>
626         </xs:sequence>
627     </xs:complexType>
628 
629     <xs:complexType name="TextOverlayType">
630         <xs:sequence>
631             <xs:element name="text" type="tns:TextRenditionType"/>
632 
633             <!-- Optional: time interval to perform overlay in -->
634             <xs:element name="interval" type="tns:TimeIntervalType" minOccurs="0"/>
635             <xs:element name="opacity" minOccurs="0" maxOccurs="1" type="xs:int"/>
636         </xs:sequence>
637     </xs:complexType>
638 
639     <xs:complexType name="ComplexJobVideoOutputType">
640         <xs:complexContent>
641             <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
↪"tns:ComplexJobOutputType">
642                 <xs:sequence>
643                     <xs:element name="scaling" minOccurs="0" maxOccurs="1" type="tns:
↪ScalingType"/>
644 
645                     <!-- Deprecated. Use <scaling> instead -->
646                     <xs:element name="resolution" type="tns:ResolutionType" minOccurs=
↪"0"/>
647 
648                     <!-- Pixel format to use, like "yuv420p", "yuv422p", "uyvy422"␣
↪etc. -->
649                     <xs:element name="pixelFormat" type="xs:string" minOccurs="0"/>
650 
651                     <xs:element name="gopSize" type="xs:unsignedShort" minOccurs="0"/>
652                     <xs:element name="maxBFrames" type="xs:unsignedShort" minOccurs="0
↪"/>
653 
654                     <!-- rc_buffer_size, size of VBV buffer -->
655                     <xs:element name="rcBufferSize" type="xs:unsignedInt" minOccurs="0
↪"/>
656 
```

```
                    <!-- rc_initial_buffer_occupancy. Should equal rc_buffer_size
                         when encoding CBR
                    -->
                    <xs:element name="rcInitialBufferOccupancy" type="xs:unsignedInt" minOccurs="0"/>

                    <!-- Minimum and maximum bitrate. Typically used for CBR output.
                         Note that for CBR, such as IMX50, you must also set
                         rcBufferSize and rcInitialOccupancy appropriately.
                         In the IMX50 case they should both be at least 2 Mbit
                         and equal.
                    -->
                    <xs:element name="minBitrate" type="xs:unsignedInt" minOccurs="0"/>
                    <xs:element name="maxBitrate" type="xs:unsignedInt" minOccurs="0"/>

                    <xs:element name="colorSiting" type="xs:unsignedShort" minOccurs="0"/>   <!-- Color Siting info for MXF output, see SMPTE 377M, E.2.35 -->
                    <xs:element name="generateThumbnails" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <!-- If set, send thumbnails to the specified URI, replacing "callback" with the licensing IP.
                                     For example, if the transcoder has been given a license by 12.34.56.78 then setting this to

                                      http://callback:8080/API/item/VX-1234/thumbnail

                                     results in a thumbnail at 0@PAL being PUT to:

                                      http://12.34.56.78:8080/API/item/VX-1234/thumbnail/0@PAL
                                -->
                                <xs:element name="uri" type="xs:anyURI" maxOccurs="unbounded"/>
                                <xs:element name="minDelay" type="tns:TimeCodeType" minOccurs="0"/>
                                <xs:element name="maxDelay" type="tns:TimeCodeType" minOccurs="0"/>
                                <xs:element name="background" type="xs:string" minOccurs="0" maxOccurs="1"/>
                                <!-- If set, use scene change detection instead of grabbing one frame every 10th second.
                                     This only has to be set - its value can be anything, even the empty string.
                                     It should really be an optional boolean, but we keep it as an optional string for backward compatibility.
                                -->
                                <xs:element name="detectorPlugin" type="xs:string" minOccurs="0"/>
                                <xs:element name="resolution" type="tns:ResolutionType" minOccurs="0"/>
                                <xs:element name="period" type="tns:TimeCodeType" minOccurs="0" maxOccurs="1"/>
                                <xs:element name="bulkyMetadataURI" type="xs:anyURI" minOccurs="0" maxOccurs="1" />
                            </xs:sequence>
```

```
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="generatePosters" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="background" type="xs:string" minOccurs="0" maxOccurs="1"/>
                                <xs:element name="resolution" type="tns:ResolutionType" minOccurs="0"/>
                                <!-- See generateThumbnails/uri -->
                                <xs:element name="uri" type="xs:anyURI" maxOccurs="unbounded"/>
                                <xs:element name="timeCode" type="tns:TimeCodeType" maxOccurs="unbounded"/>
                                <xs:element name="format" type="xs:string" minOccurs="0" maxOccurs="1"/>
                                <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded"/>
                            </xs:sequence>
                            <xs:attribute name="smallPosters" type="xs:boolean" use="optional"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="detectFaces" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="metadataUri" type="xs:anyURI"/>
                                <xs:element name="faceDetectorPlugin" type="xs:string"/>    <!-- maps to TranscoderConfigurationDocument/faceDetectorPlugin/alias -->
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="overlay" type="tns:OverlayType" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="textOverlay" type="tns:TextOverlayType" minOccurs="0" maxOccurs="unbounded"/>

                    <!-- If set, strip SPS/PPS from packets before handing them off to the muxer.
                         This is required when muxing AVC-Intra for Avid Media Composer, and possibly FCP.
                    -->
                    <xs:element name="stripParameterSets" type="xs:boolean" minOccurs="0"/>

                    <!-- If set, add SPS/PPS to the first packet output by the demuxer for this stream.
                         The data is taken from extradata - set extradata as well if it is not
                         present or if it is incorrect
                    -->
                    <xs:element name="addParameterSets" type="xs:boolean" minOccurs="0"/>

                    <!-- If set, explicitly use this for the parameter set data.
                         If not set, then whatever the demuxer reports as extradata for this stream will be used.
                    -->
```

```
                    <xs:element name="parameterSets" type="xs:hexBinary" minOccurs="0
↪"/>

                    <!-- If set, use this level.
                         For H.264, the value of this should be ten times the decimal␣
↪value of the level.
                         In other words, 1.0 -> 10, 5.1 -> 51 etc.
                         This applies for both libx264 and MainConcept.
                         If used, then profile should also be set (via <preset>).
                    -->
                    <xs:element name="level" type="xs:int" minOccurs="0"/>

                    <!-- Deprecated -->
                    <xs:element name="disableFrameDupDrop" type="xs:boolean"␣
↪minOccurs="0" />

                    <!-- Use <forceCFR> to force the output to be CFR, according to
↪<timeBase>.
                         For instance, if timeBase = 1/25 then the transcoder will␣
↪take all frames
                         from the input and make all of them 40 ms long.
                         This is useful for formats where the first few frames have␣
↪bad durations,
                         or where the framerate of the entire file is demonstrably␣
↪wrong.
                         Not used when remuxing.
                    -->
                    <xs:element name="forceCFR" type="xs:boolean" minOccurs="0" />

                    <!-- If true, burn the input file's timecode in the output -->
                    <xs:element name="burnTimecode" type="xs:boolean" minOccurs="0" />

                    <!-- Only used for image transcodes -->
                    <xs:element name="imageQuality" type="xs:integer" minOccurs="0" />

                    <!-- MXF-ish display rectangle.
                         This means values straight from CDCIDescriptor for MXF,
                         but scaled down by SAR for clap in MOV.
                         In other words, "to display" space, not "displayed" (aka␣
↪raster) space.
                    -->
                    <xs:element name="displayWidth" type="tns:RationalType" minOccurs=
↪"0"/>
                    <xs:element name="displayHeight" type="tns:RationalType"␣
↪minOccurs="0"/>
                    <xs:element name="displayXOffset" type="tns:RationalType"␣
↪minOccurs="0"/>
                    <xs:element name="displayYOffset" type="tns:RationalType"␣
↪minOccurs="0"/>

                    <!-- DAR = displayWidth/displayHeight * containerSAR -->
                    <xs:element name="containerSAR" type="tns:AspectRatioType"␣
↪minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

```

```
    <!--
        ScalingType

        Used to set cropping and scaling parameters to the transcoder.

        By default, the transcoder will attempt to maintain the display aspect
        ratio (DAR) of the cropped input. Use targetDAR to specify a different
        DAR to maintain.

        The transcoder will typically try to adjust the PAR so that the cropped
        picture ends up with the correct DAR. This minimizes the amount of
        processing required. Use pixelAspectRatio to set the PAR explicitly, in
        which case either width or height will be adjusted to maintain DAR.

        Use width and height to scale in those dimensions. If only one of them
        is set and PAR is set, the other one will be adjusted so the result
        matches the target DAR. If both are set and and PAR is set, the
        transcoder will take them as is.

        Setting neither width nor height while PAR is set results in undefined
        behavior.

        The transcoder will always double-check the resulting dimensions and PAR
        against the desired DAR. If there's a mismatch, the job will fail. If
        you want to force the transcoder to accept your settings, set targetDAR
        manually to the resulting DAR.
    -->
    <xs:complexType name="ScalingType">
        <xs:sequence>
            <xs:element name="width" minOccurs="0" maxOccurs="1" type="xs:unsignedInt
↪"/>
            <xs:element name="height" minOccurs="0" maxOccurs="1" type="xs:unsignedInt
↪"/>

            <!-- Specifies the number of pixels to crop out of each side.
                 Be careful when cropping odd numbers of pixels in any dimension
                 that is subsampled. For instance, cropping an odd number of
                 lines in YUV 4:2:0. This will cause the chroma siting to shift.
            -->
            <xs:element name="top" minOccurs="0" maxOccurs="1" type="xs:int"/>
            <xs:element name="bottom" minOccurs="0" maxOccurs="1" type="xs:int"/>
            <xs:element name="left" minOccurs="0" maxOccurs="1" type="xs:int"/>
            <xs:element name="right" minOccurs="0" maxOccurs="1" type="xs:int"/>
            <xs:element name="padColor" minOccurs="0" maxOccurs="1" type="xs:string"/>
↪ <!-- HTML (#rrggbb), if crop is negative -->

            <!-- Specifies rotation. If no resolution/PAR is set, the PAR
                 and resolution of the input is inverted to (hopefully)
                 result in a pixel-to-pixel accurate flipping.
                 Valid values are "left", "right" and "upsidedown" for 90, -90 and
                 180 degrees rotation, respectively
            -->
            <xs:element name="rotate" minOccurs="0" maxOccurs="1" type="xs:string"/>

            <!-- PAR -->
            <xs:element name="pixelAspectRatio" minOccurs="0" maxOccurs="1" type="tns:
↪AspectRatioType"/>

```

```
            <!-- Desired display aspect ratio -->
            <xs:element name="targetDAR" minOccurs="0" maxOccurs="1" type="tns:AspectRatioType"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ComplexJobSubtitleOutputType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:ComplexJobOutputType">
                <xs:sequence/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="BorderType">
        <!-- Alpha + RGB color of border -->
        <xs:attribute name="a" type="xs:unsignedByte" use="required"/>
        <xs:attribute name="r" type="xs:unsignedByte" use="required"/>
        <xs:attribute name="g" type="xs:unsignedByte" use="required"/>
        <xs:attribute name="b" type="xs:unsignedByte" use="required"/>

        <!-- Width of border in pixels in display space -->
        <xs:attribute name="width" type="xs:unsignedByte" use="required"/>
    </xs:complexType>

    <xs:complexType name="TransitionType">
        <xs:sequence>
            <xs:element name="duration" type="tns:TimeCodeType"/>
            <xs:choice>
                <!-- SMPTE wipe code (see S258m) -->
                <xs:element name="wipe" type="xs:int"/>

                <!-- Other transition, like "CrossDissolve". Corresponds to Fabric's transitionSubTypeType -->
                <xs:element name="transition" type="xs:string"/>
            </xs:choice>
            <xs:element name="horizRepeat" type="xs:int" minOccurs="0"/>
            <xs:element name="vertRepeat" type="xs:int" minOccurs="0"/>

            <!-- startPercentage and endPercentage can optionally be used to override the normal 0-100% transition range -->
            <xs:element name="startPercentage" type="xs:int" minOccurs="0"/>
            <xs:element name="endPercentage" type="xs:int" minOccurs="0"/>

            <!-- If set and true, reverse the direction of the wipe -->
            <xs:element name="reverse" type="xs:boolean" minOccurs="0"/>

            <xs:element name="border" type="tns:BorderType" minOccurs="0"/>
            <xs:element name="color" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Generic "I want this stream from that input" type. Works for both audio and video -->
    <xs:complexType name="ComplexJobInputType">
        <xs:sequence>
            <xs:element name="id" type="xs:int"/>
```

```
            <xs:element name="stream" type="xs:unsignedShort"/>

            <!-- Optional: transition effect to use when this input is followed by another input in a timeline -->
            <xs:element name="transition" type="tns:TransitionType" minOccurs="0"/>

            <!-- Optional: Use this to set settings for decoders -->
            <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <!--
          Input type for grabbing a specific channel from an input audio stream.
          Several of these as inputs in a connection make it possible to create a new stream from M existing channels in N input streams.
    -->
    <xs:complexType name="ComplexJobAudioChannelMapInputType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:ComplexJobInputType">
                <xs:sequence>
                    <xs:element name="channel" type="xs:unsignedShort"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Input type for joining together several mono audio sequences
         In other words, this type defines a mono timeline.
         The mono intervals specified by each ComplexJobAudioChannelMapInputType
         element in this type are joined together to produce the output.

         Several ComplexJobAudioChannelSequenceInputType elements are used in
         ComplexJobType to produce a full timeline - one for each channel.

         Note that if no channels need to be extracted separately a
         ComplexJobInputType array should be use instead (see
         ComplexJobType/connection/input).
    -->
    <xs:complexType name="ComplexJobAudioChannelSequenceInputType">
        <xs:sequence>
            <xs:element name="input" type="tns:ComplexJobAudioChannelMapInputType" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ComplexJobMixType">
        <xs:sequence>
            <xs:element name="effect" type="tns:EffectType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:int" use="required"/>
        <xs:attribute name="stream" type="xs:unsignedShort" use="required"/>
        <xs:attribute name="channel" type="xs:unsignedShort" use="required"/>
        <xs:attribute name="gain" type="xs:float" use="required"/>              <!-- linear; 1.0 = 0 dB etc. -->
    </xs:complexType>
```

```
941
942     <xs:complexType name="ComplexJobMixInputType">
943        <xs:sequence>
944             <xs:element name="mix" type="tns:ComplexJobMixType" minOccurs="1"␣
↪maxOccurs="unbounded"/>
945         </xs:sequence>
946     </xs:complexType>
947
948     <!-- Used to request extraction of various types of metadata from input files to␣
↪be reported to a resource as BulkyMetadata -->
949     <xs:complexType name="ComplexJobBulkyMetadataRequestType">
950         <xs:sequence>
951             <xs:element name="targetUri" type="xs:anyURI"/>
↪       <!-- URI to write BulkyMetadata to -->                                    ␣
952             <xs:element name="failIfTimecodeNotPresent" type="xs:boolean" minOccurs="0
↪"/>   <!-- If true, fail the job if we didn't find a single timecode -->
953         </xs:sequence>
954     </xs:complexType>
955
956     <!-- Information needed by the PartialFileDemuxer -->
957     <xs:complexType name="PartialFileDemuxerInfoType">
958         <xs:sequence>
959             <!-- The user can either use a full desriptor or give a URI pointing to␣
↪one -->
960             <xs:choice>
961                 <xs:element name="descriptor" type="tns:PartialFileDescriptorType"/>
962                 <xs:element name="descriptorLocation" type="xs:anyURI"/>
963             </xs:choice>
964
965             <!-- Offset into original file that ComplexJobType/input/uri
966                  consists of. In other words, how far into the file the binary
967                  blob was cut.
968             -->
969             <xs:element name="byteOffset" type="xs:long" minOccurs="0"/>
970             <xs:element name="adjustForPTSPredecessors" type="xs:boolean" minOccurs="0
↪" />
971         </xs:sequence>
972     </xs:complexType>
973
974     <xs:complexType name="ComplexJobAtomType">
975         <xs:attribute name="uri" type="xs:anyURI" use="required"/>
976
977         <!-- generated if not set -->
978         <xs:attribute name="sourcePackageID" type="tns:UMIDType" use="optional"/>
979     </xs:complexType>
980
981     <xs:complexType name="AnalyzeAudioChannelType">
982         <xs:sequence>
983             <xs:element name="tone" type="xs:float" minOccurs="0" maxOccurs="unbounded
↪"/>
984         </xs:sequence>
985
986         <!-- Which channel in which stream to use these tones and thresholds for -->
987         <xs:attribute name="stream" type="xs:unsignedShort" use="optional"/>
988         <xs:attribute name="channel" type="xs:unsignedShort" use="optional"/>
989
990         <!-- Silence threshold. Linear value proportional to maximum sample value.
991              In other words:
```

```
992                 0.0   = -inf dB
993                 0.001=  -30 dB (default)
994                 1.0   =    0 dB
995         -->
996         <xs:attribute name="thresh" type="xs:float" use="optional"/>
997     </xs:complexType>
998
999     <xs:complexType name="AnalyzeAudioType">
1000         <xs:sequence>
1001             <xs:element name="otif" type="tns:ComplexJobOTIFType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
1002         </xs:sequence>
1003     </xs:complexType>
1004
1005     <xs:complexType name="AnalyzeVideoType">
1006         <xs:sequence>
1007             <xs:element name="otif" type="tns:ComplexJobOTIFType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
1008         </xs:sequence>
1009     </xs:complexType>
1010
1011     <xs:complexType name="HighlighterType">
1012         <xs:sequence>
1013             <xs:element name="model" type="xs:string" minOccurs="1" maxOccurs="1"/>
1014         </xs:sequence>
1015     </xs:complexType>
1016
1017     <xs:complexType name="SmartCropType">
1018         <xs:sequence>
1019             <xs:element name="aspect" type="xs:string" minOccurs="1" maxOccurs="1"/>
1020         </xs:sequence>
1021     </xs:complexType>
1022
1023     <!-- Used for analyzing input video. See transcoder tickets #82 and #83 -->
1024     <xs:complexType name="ComplexJobAnalyzeType">
1025         <xs:sequence>
1026             <xs:element name="channel" type="tns:AnalyzeAudioChannelType" minOccurs="0
↪" maxOccurs="unbounded"/>
1027             <xs:element name="audio" type="tns:AnalyzeAudioType" minOccurs="0"␣
↪maxOccurs="1"/>
1028             <xs:element name="video" type="tns:AnalyzeVideoType" minOccurs="0"␣
↪maxOccurs="1"/>
1029             <xs:element name="highlighter" type="tns:HighlighterType" minOccurs="0"␣
↪maxOccurs="1"/>
1030             <xs:element name="smartcrop" type="tns:SmartCropType" minOccurs="0"␣
↪maxOccurs="1"/>
1031         </xs:sequence>
1032         <xs:attribute name="metadataUri" type="xs:anyURI" use="required"/>
1033
1034         <!-- Thresholds are relative to the maximum pixel value, meaning values␣
↪around 0-0.1 are reasonable -->
1035         <xs:attribute name="blackThresh" type="xs:float" use="optional"/>
1036         <xs:attribute name="blackPercentage" type="xs:int" use="optional"/>
1037         <xs:attribute name="barsThresh" type="xs:float" use="optional"/>
1038         <xs:attribute name="barsPercentage" type="xs:int" use="optional"/>
1039         <xs:attribute name="freezeThresh" type="xs:float" use="optional"/>
1040         <xs:attribute name="freezeTime" type="xs:float" use="optional"/>
1041     </xs:complexType>
```

```
    <!-- Used for pairing a resource with a name in the OTIF type -->
    <xs:complexType name="NameURIPairType">
        <xs:sequence>
            <xs:element name="name" minOccurs="1" maxOccurs="1" type="xs:string"/>
            <xs:element name="uri" minOccurs="1" maxOccurs="1" type="xs:anyURI"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="TranscoderJobType">
        <xs:sequence>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ComplexJobOutputFormatType">
        <xs:sequence>
            <xs:element name="id" type="xs:int"/>

            <xs:choice>
                <!-- Normal single-file output -->
                <xs:element name="uri" type="xs:anyURI"/>

                <xs:element name="range" type="tns:SequenceRangeType"/>
                <!-- For multi-OPAtom output (mxf_multiatom*)
                    The number of connections to the muxer must equal the number of␣
↪atom elements here
                -->
                <xs:element name="atom" type="tns:ComplexJobAtomType" maxOccurs=
↪"unbounded"/>
            </xs:choice>

            <xs:element name="containerFormat" type="xs:string"/>
            <xs:element name="overlay" type="tns:OverlayType" minOccurs="0" maxOccurs=
↪"unbounded"/>
            <xs:element name="textOverlay" type="tns:TextOverlayType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="dms1Source" minOccurs="0">
                <xs:complexType>
                    <xs:choice>
                        <xs:element name="demuxerId" type="xs:int"/>
                        <xs:element name="metadata" type="tns:DMS1Type"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <!-- DEPRECATED: String representation of the SMPTE 12M time code to use␣
↪for the first frame -->
            <xs:element name="initialSMPTETimecode" type="xs:string" minOccurs="0"/>

            <!-- Corresponds to StartTimecode in TimecodeComponent in MXF -->
            <xs:element name="startTimecode" type="xs:long" minOccurs="0"/>

            <!-- Corresponds to RoundedTimeBase in TimecodeComponent in MXF -->
            <xs:element name="roundedTimeBase" type="xs:int" minOccurs="0"/>

            <!-- Corresponds to DropFrame in TimecodeComponent in MXF -->
            <xs:element name="dropFrame" type="xs:boolean" minOccurs="0"/>

            <!-- Set to true if muxing MOV and the user wants the job to fail if any␣
↪stream lacks an estimate for numberOfPackets -->
```

```
            <xs:element name="requireFaststart" type="xs:boolean" minOccurs="0"/>
            <!-- Set to desired bitrate for CBR muxing: Mainly used for mpegts -->
            <xs:element name="muxrate" type="xs:unsignedInt" minOccurs="0"/>

            <!-- If using a muxer that supports outputing a␣
↪PartialFileDescriptorDocument,
                 setting this causes the resulting document to be written
                 to the specified URI when the job finishes.
            -->
            <xs:element name="pfdTargetUri" type="xs:anyURI" minOccurs="0"/>

            <!-- Material and tape packages to use in the file -->
            <xs:element name="mxfPackages" type="tns:MXFPackagesType" minOccurs="0"/>

            <!-- Global flat metadata -->
            <xs:element name="metadata" type="tns:KeyValuePairType" minOccurs="0"␣
↪maxOccurs="unbounded"/>

            <!-- MaterialPackage -> Name for MXF.
                 Not applicable to most other formats (except maybe MOV).
            -->
            <xs:element name="clipName" type="xs:string" minOccurs="0"/>

            <!-- Controls how the maximum time period that each chunk of samples is␣
↪going to be, only used for output of QuickTime files (MOV/MP4) -->
            <xs:element name="maxChunkDuration" type="tns:TimeCodeType" minOccurs="0"/
↪>

            <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*ComplexJobDocument*

```
    <xs:element name="ComplexJobDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:ComplexJobType" />
```

*ImageJobDocument*

```
    <xs:element name="ImageJobDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:ComplexJobType" />
    <xs:complexType name="ComplexJobType">
        <xs:complexContent>
            <xs:extension base="tns:TranscoderJobType">
        <xs:sequence>
            <xs:element name="input" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="xs:int"/>

                        <!-- Use multiple URIs if transcoding image sequences -->
                        <xs:element name="uri" type="xs:anyURI" maxOccurs="unbounded"/
↪>

                        <xs:element name="range" type="tns:SequenceRangeType"␣
↪maxOccurs="unbounded"/>

```

```
                        <xs:element name="partialFile" type="tns:PartialFileDemuxerInfoType" minOccurs="0"/>
                        <xs:element name="interval" type="tns:TimeIntervalType" minOccurs="0"/>

                        <!-- If set to true, then interval applies to DTS, not PTS.
                             This may be useful if remuxing and the input file lacks PTS:es.
                             We almost always want to filter frames/packets on PTS though.
                             Use of this is discouraged for now.
                        -->
                        <xs:element name="intervalIsDts" type="xs:boolean" minOccurs="0"/>

                        <xs:element name="dms1TargetUri" type="xs:anyURI" minOccurs="0"/>
                        <!-- faststartDuration is needed if muxing MOV and faststart is desired and the number of packets for each stream in the input is unknown or wrong.
                             The transcoder will make an estimate for the number of packets and the muxer will use that to reserve space in the header for the moov tag.
                             Set this value to the length of the input if known through some other means or failing that, set it to a high value like ten hours.

                             The override attribute should be set to true if the transcoder is wrong regarding the duration of the input or the number of packets for some stream.
                        -->
                        <xs:element name="faststartDuration" minOccurs="0">
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:extension base="tns:TimeCodeType">
                                        <xs:attribute name="override" type="xs:boolean" use="required"/>
                                    </xs:extension>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="bulkyMetadataRequest" type="tns:ComplexJobBulkyMetadataRequestType" minOccurs="0"/>
                        <!-- Both of these elemets are hacks to handle broken files. The integer is the index of the videostream to which the hack should be applied. -->
                        <xs:element name="scanForStartPTS" type="xs:int" minOccurs="0" />
                        <xs:element name="doubleDurationHack" type="xs:int" minOccurs="0" />

                        <!-- Optional: Use this to set settings for demuxers -->
                        <xs:element name="demuxerSetting" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded"/>

                        <xs:element name="analyze" type="tns:ComplexJobAnalyzeType" minOccurs="0"/>
                        <!-- Use for setting a page number (in PDFs) -->
                        <xs:element name="pageno" type="xs:int" minOccurs="0" maxOccurs="1"/>
                    </xs:sequence>
```

```
                </xs:complexType>
            </xs:element>
            <xs:element name="output" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" base="tns:ComplexJobOutputFormatType"/>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="connection" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:choice>
                            <!-- Use of more than one input means that the streams␣
↪should be concatenated -->
                            <xs:element name="input" type="tns:ComplexJobInputType"␣
↪minOccurs="1" maxOccurs="unbounded"/>
                            <!-- Used to extract and interleave channels from␣
↪multiple input streams into this audio stream -->
                            <xs:element name="audioChannelMapInput" type="tns:
↪ComplexJobAudioChannelMapInputType" minOccurs="1" maxOccurs="unbounded"/>
                            <!-- Used to interleave several mono timelines into this␣
↪audio stream -->
                            <xs:element name="audioChannelSequenceInput" type="tns:
↪ComplexJobAudioChannelSequenceInputType" minOccurs="1" maxOccurs="unbounded"/>

                            <!-- Used to mix several mono streams into one multi-
↪channel stream
                                 Each audioMixInput element specifies the mix matrix␣
↪for one mono channel in this stream -->
                            <xs:element name="audioMixInput" type="tns:
↪ComplexJobMixInputType" minOccurs="1" maxOccurs="unbounded"/>
                        </xs:choice>
                        <xs:choice>
                            <xs:element name="audioOutput" type="tns:
↪ComplexJobAudioOutputType"/>
                            <xs:element name="videoOutput" type="tns:
↪ComplexJobVideoOutputType"/>
                            <xs:element name="subtitleOutput" type="tns:
↪ComplexJobSubtitleOutputType"/>
                        </xs:choice>
                        <!-- PID, or similar per-stream ID for use by the muxer -->
                        <xs:element name="pid" type="xs:int" minOccurs="0"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:extension>
    </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="XMPReadOperation">
        <xs:sequence>
            <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="uri" type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
            <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
```

```
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="XMPWriteOperation">
        <xs:sequence>
            <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="uri" type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
            <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0"
↪maxOccurs="unbounded"/>
            <xs:element name="xmp" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*XMPJobDocument*

```
    <xs:element name="XMPJobDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:XMPJobType" />
    <xs:complexType name="XMPJobType">
        <xs:complexContent>
            <xs:extension base="tns:TranscoderJobType">
                <xs:sequence>
                    <xs:element name="read" type="tns:XMPReadOperation" minOccurs="0"
↪maxOccurs="unbounded" />
                    <xs:element name="write" type="tns:XMPWriteOperation" minOccurs="0
↪" maxOccurs="unbounded" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TransferOperation">
        <xs:sequence>
            <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="sourceUri" type="xs:anyURI" minOccurs="1" maxOccurs=
↪"unbounded"/>
            <xs:element name="destinationUri" type="xs:anyURI" minOccurs="1"
↪maxOccurs="unbounded"/>
            <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0"
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*TransferJobDocument*

```
    <xs:element name="TransferJobDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:TransferJobType" />
    <xs:complexType name="TransferJobType">
        <xs:complexContent>
            <xs:extension base="tns:TranscoderJobType">
                <xs:sequence>
                    <xs:element name="transfer" type="tns:TransferOperation"
↪minOccurs="0" maxOccurs="unbounded" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="HashComputeOperation">
```

```
        <xs:sequence>
            <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="uri" type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
            <xs:element name="function" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*HashJobDocument*

```
    <xs:element name="HashJobDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:HashJobType" />
    <xs:complexType name="HashJobType">
        <xs:complexContent>
            <xs:extension base="tns:TranscoderJobType">
                <xs:sequence>
                    <xs:element name="compute" type="tns:HashComputeOperation"␣
↪minOccurs="0" maxOccurs="unbounded" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

*ShapeDeductionJobDocument*

```
    <xs:element name="ShapeDeductionJobDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:ShapeDeductionJobType" />
    <xs:complexType name="ShapeDeductionJobType">
        <xs:complexContent>
            <xs:extension base="tns:TranscoderJobType">
        <xs:sequence>
            <xs:element name="input" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
                        <xs:element name="uri" type="xs:anyURI" minOccurs="0"␣
↪maxOccurs="1"/>
                        <xs:element name="range" type="tns:SequenceRangeType"␣
↪maxOccurs="unbounded"/>
                        <xs:element name="image" type="xs:boolean" minOccurs="0"␣
↪maxOccurs="1"/>
                        <xs:element name="useMediaInfo" type="xs:boolean" minOccurs="0
↪" maxOccurs="1"/>
                        <xs:element name="verbose" type="xs:boolean" minOccurs="0"␣
↪maxOccurs="1"/>
                        <xs:element name="setting" type="tns:KeyValuePairType"␣
↪minOccurs="0" maxOccurs="unbounded"/>
                        <!-- Use for setting a page number (in PDFs) -->
                        <xs:element name="pageno" type="xs:int" minOccurs="0"␣
↪maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
            </xs:extension>
        </xs:complexContent>
```

```
    </xs:complexType>
```

*DurationJobDocument*

```
    <xs:element name="DurationJobDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:DurationJobType" />
    <xs:complexType name="DurationJobType">
        <xs:complexContent>
            <xs:extension base="tns:TranscoderJobType">
        <xs:sequence>
            <xs:element name="input" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="uri" type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Types used for defining a MOV index generation job -->
```

*MOVIndexJobDocument*

```
    <xs:element name="MOVIndexJobDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:MOVIndexJobType"/>
    <xs:complexType name="MOVIndexJobType">
        <xs:complexContent>
            <xs:extension base="tns:TranscoderJobType">
        <xs:sequence>
            <xs:element name="targetUri" type="xs:anyURI"/>         <!-- http://example.com/index.mov -->
            <xs:element name="source" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="uri" type="xs:anyURI"/>   <!-- absolute URI to derive index from. example: http://example.com/essence/video.m2v -->
                        <xs:element name="alias" type="xs:anyURI" minOccurs="0"/> <!-- relative URI to write to file. example:     essence/video.m2v -->
                        <xs:element name="absoluteAlias" type="xs:anyURI" minOccurs="0"/> <!-- absolute URI to write to file. example:     /mnt/something/essence/video.m2v -->
                        <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

*MXFTimecodeExtractionJobDocument*

```
    <xs:element name="MXFTimecodeExtractionJobDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:MXFTimecodeExtractionJobType"/>
    <xs:complexType name="MXFTimecodeExtractionJobType">
        <xs:complexContent>
            <xs:extension base="tns:TranscoderJobType">
        <xs:sequence>
            <xs:element name="sourceUri" type="xs:anyURI"/>         <!-- URI to read MXF from -->
            <xs:element name="targetUri" type="xs:anyURI"/>         <!-- URI to write BulkyMetadata to -->
        </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Generates an Op1b MXF file that points to several pieces of external essence -->
```

*MXFOp1bJobDocument*

```
    <xs:element name="MXFOp1bJobDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:MXFOp1bJobType"/>
    <xs:complexType name="MXFOp1bJobType">
        <xs:complexContent>
            <xs:extension base="tns:TranscoderJobType">
        <xs:sequence>
            <xs:element name="output" type="xs:anyURI"/>        <!-- URI to write to -->
            <xs:element name="reference" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <!-- URI to read metadata from. Usually equal to locator[0] -->
                        <xs:element name="source" type="xs:anyURI"/>
                        <!-- List of stream the user wants to include in this reference -->
                        <xs:element name="stream" type="xs:unsignedShort" maxOccurs="unbounded"/>
                        <!--
                             locator: List of network locators (URIs) to external essence, ordered by preference.
                                      Relative URIs should go first, absolute URIs as fallbacks near the end.
                                 Ex.: <locator>essence/video.m2v</locator>
                                      <locator>ftp://example.com/essence/video.m2v</locator>
                        -->
                        <xs:element name="locator" type="xs:anyURI" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

*SegmentationJobDocument*

```
    <xs:element name="SegmentationJobDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SegmentationJobType"/>
    <xs:complexType name="SegmentationJobType">
        <xs:complexContent>
            <xs:extension base="tns:TranscoderJobType">
        <xs:sequence>
            <!-- URI to material to segment. example: http://example.com/video.ts -->
            <xs:element name="input" type="xs:anyURI"/>
            <!-- URI to write playlist to when done. example: http://example.com/foo.m3u -->
            <xs:element name="playlistOutput" type="xs:anyURI"/>
            <!-- The prefix and postfix combine with a number to form the full URI. example:

                    prefix = "http://example.com/media/segment"
                  postfix = ".ts"
                segment 1 = http://example.com/media/segment1.ts
                segment 2 = http://example.com/media/segment2.ts etc.
            -->
            <xs:element name="segmentUriPrefix" type="xs:string"/>
            <xs:element name="segmentUriPostfix" type="xs:string"/>
            <!-- Container format to use for all segments. example: "mpegts" -->
            <xs:element name="containerFormat" type="xs:string"/>
            <!-- Suggested length of each segment. The transcoder will do its best if this is not a multiple of the GOP length -->
            <xs:element name="segmentLength" type="tns:TimeCodeType"/>
        </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

*AAFGeneratorJobDocument*

```
    <xs:element name="AAFGeneratorJobDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:AAFGeneratorJobType" />
    <xs:complexType name="AAFGeneratorJobType">
        <xs:complexContent>
            <xs:extension base="tns:TranscoderJobType">
                <xs:sequence>
                    <xs:choice>
                        <xs:element name="aaf" type="xs:hexBinary" />
                        <xs:element name="xml" type="xs:string" />
                    </xs:choice>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- XML types for NLEJob -->

    <xs:complexType name="SubClipType">
        <xs:attribute name="id" type="xs:int" use="required"/>
        <xs:attribute name="start" type="xs:int" use="required"/>
        <xs:attribute name="length" type="xs:unsignedInt" use="required"/>
    </xs:complexType>

    <xs:complexType name="ClipType">
```

```
1443        <xs:sequence>
1444            <xs:element name="subClip" type="tns:SubClipType" minOccurs="0" maxOccurs=
    ↪"unbounded"/>
1445        </xs:sequence>
1446        <xs:attribute name="uri" type="xs:anyURI" use="optional"/>
1447        <xs:attribute name="stream" type="xs:unsignedShort" use="optional"/>
1448        <xs:attribute name="id" type="xs:anyURI" use="optional"/>
1449        <xs:attribute name="track" type="xs:unsignedShort" use="optional"/>
1450    </xs:complexType>
1451
1452    <xs:complexType name="VideoClipType">
1453        <xs:complexContent>
1454            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
    ↪"tns:ClipType">
1455                <xs:attribute name="directCopy" type="xs:boolean" use="optional"/>
1456            </xs:extension>
1457        </xs:complexContent>
1458    </xs:complexType>
1459
1460    <xs:complexType name="AudioClipType">
1461        <xs:complexContent>
1462            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
    ↪"tns:ClipType">
1463                <xs:attribute name="channel" type="xs:unsignedShort" use="required"/>
1464            </xs:extension>
1465        </xs:complexContent>
1466    </xs:complexType>
1467
1468    <xs:complexType name="TextRenditionType">
1469        <xs:sequence>
1470            <xs:element name="line" type="xs:string" maxOccurs="unbounded"/>
1471        </xs:sequence>
1472        <xs:attribute name="align" type="xs:string" use="optional"/>        <!-- Text␣
    ↪alignment. May be "center" (default), "left" or "right" -->
1473        <xs:attribute name="x" type="xs:int" use="optional"/>               <!--␣
    ↪default = 0. Center of text box relative to the center of the frame, in pre-SAR␣
    ↪pixels  (X part) -->
1474        <xs:attribute name="y" type="xs:int" use="optional"/>               <!--␣
    ↪default = 0. Center of text box relative to the center of the frame, in pre-SAR␣
    ↪pixels  (Y part) -->
1475        <xs:attribute name="xRel" type="xs:double" use="optional"/>              <!--
    ↪ optional, overrides x if set, in units of full width -->
1476        <xs:attribute name="yRel" type="xs:double" use="optional"/>              <!--
    ↪ optional, overrides y if set, in units of full height -->
1477        <xs:attribute name="horizontalBase" type="xs:string" use="optional"/>    <!--
    ↪ optional, only used if xRel is set, specifies if corner specifies left of test box,
    ↪ center (default), or right -->
1478        <xs:attribute name="verticalBase" type="xs:string" use="optional"/>      <!--
    ↪ optional, only used if yRel is set, specifies if corner specifies top of test box,␣
    ↪middle (default), or bottom -->
1479        <xs:attribute name="font" type="xs:string" use="optional"/>        <!--␣
    ↪default = "Arial". Font to use -->
1480        <xs:attribute name="size" type="xs:unsignedInt" use="optional"/>   <!--␣
    ↪default = 12. Size of font in pixels -->
1481        <xs:attribute name="sizeRel" type="xs:double" use="optional"/>   <!--␣
    ↪optional, overrides size if set, size of font in units of full height -->
1482        <xs:attribute name="r" type="xs:unsignedByte" use="optional"/>     <!--␣
    ↪default = 255. Text color, red -->
```

```
        <xs:attribute name="g" type="xs:unsignedByte" use="optional"/>      <!-- default = 255. Text color, green -->
        <xs:attribute name="b" type="xs:unsignedByte" use="optional"/>      <!-- default = 255. Text color, blue -->
        <xs:attribute name="a" type="xs:unsignedByte" use="optional"/>      <!-- default = 255. Text color, alpha -->

        <!-- Outline type. Values:
               "none" = No outline (default)
                "box" = Draw solid box behind text, SVT style
             "stroke" = Stroke text
        -->
        <xs:attribute name="outline" type="xs:string" use="optional"/>

        <!--    "box": How much larger the box is on each side compared to the text's bounding box
             "stroke": How far out the font is stroked
        -->
        <xs:attribute name="outlineSize" type="xs:unsignedInt" use="optional"/>

        <xs:attribute name="outlineR" type="xs:unsignedByte" use="optional"/>      <!- default = 0. Outline color, red -->
        <xs:attribute name="outlineG" type="xs:unsignedByte" use="optional"/>      <!- default = 0. Outline color, green -->
        <xs:attribute name="outlineB" type="xs:unsignedByte" use="optional"/>      <!- default = 0. Outline color, blue -->
        <xs:attribute name="outlineA" type="xs:unsignedByte" use="optional"/>      <!- default = 255. Outline color, alpha -->
        <xs:attribute name="language" type="xs:string" use="optional"/>            <!- ISO-639 -->
    </xs:complexType>

    <xs:complexType name="SubtitleClipType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:TextRenditionType">
                <xs:attribute name="id" type="xs:int" use="required"/>
                <xs:attribute name="length" type="xs:unsignedInt" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Like TransitionType, except we use xs:unsignedInt for duration and only attributes -->
    <xs:complexType name="NLEJobTransitionType">
        <xs:sequence>
            <xs:element name="border" type="tns:BorderType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="length" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="wipe" type="xs:int" use="optional"/>
        <xs:attribute name="transition" type="xs:string" use="optional"/>
        <xs:attribute name="horizRepeat" type="xs:int" use="optional"/>
        <xs:attribute name="vertRepeat" type="xs:int" use="optional"/>
        <xs:attribute name="startPercentage" type="xs:int" use="optional"/>
        <xs:attribute name="endPercentage" type="xs:int" use="optional"/>
        <xs:attribute name="reverse" type="xs:boolean" use="optional"/>

        <xs:attribute name="color" type="xs:string" use="optional"/>
```

```
        <!-- For internal transcoder use when cutting NLEJobDocuments -->
        <xs:attribute name="delta" type="xs:unsignedInt" use="optional"/>
    </xs:complexType>

    <xs:complexType name="TrackSegmentType">
        <xs:sequence>
            <xs:element name="effect" type="tns:EffectType" minOccurs="0" maxOccurs=
↪"unbounded"/>
            <xs:element name="transition" type="tns:NLEJobTransitionType" minOccurs="0
↪"/>
        </xs:sequence>
        <xs:attribute name="fillerLength" type="xs:unsignedInt" use="optional"/>    <!
↪-- filler, with the value being the length to fill -->
        <xs:attribute name="subClip" type="xs:int" use="optional"/>                 <!
↪-- ID of subClip -->
    </xs:complexType>

    <xs:complexType name="EffectPointType">
        <xs:attribute name="value" type="xs:float" use="required"/>

        <!-- The position of this value relative to the segment -->
        <xs:attribute name="position" type="xs:int" use="required"/>
    </xs:complexType>

    <xs:complexType name="EffectParameterType">
        <xs:sequence>
            <!-- Points are used for changing a parameter's value over time -->
            <xs:element name="point" type="tns:EffectPointType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>

        <!-- Setting this value causes it to be applied to the segment's entire␣
↪interval.
             In other words, it makes the parameter non-temporal.
        -->
        <xs:attribute name="value" type="xs:float" use="optional"/>
    </xs:complexType>

    <xs:complexType name="EffectType">
        <xs:sequence>
            <xs:element name="timeBase" type="tns:TimeBaseType" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="parameter" type="tns:EffectParameterType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>


    <xs:complexType name="TrackType">
        <xs:sequence>
            <xs:element name="segment" type="tns:TrackSegmentType" maxOccurs=
↪"unbounded"/>
        </xs:sequence>
    </xs:complexType>

```

```
    <xs:complexType name="NLEJobSequenceType">
        <xs:sequence>
            <xs:element name="track" type="tns:TrackType" maxOccurs="unbounded"/>   <!-- List of tracks in ascending priority -->
        </xs:sequence>
        <xs:attribute name="color" type="xs:string" use="optional"/>
        <xs:attribute name="id" type="xs:int" use="required"/>
        <xs:attribute name="length" type="xs:unsignedInt" use="required"/>
    </xs:complexType>

    <xs:complexType name="NLEJobVideoOutputType">
        <xs:sequence>
            <xs:element name="preset" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="uri" type="xs:anyURI" use="optional"/>                  <!-- used when outputing OPAtom -->
        <xs:attribute name="sequence" type="xs:int" use="required"/>
        <xs:attribute name="codec" type="xs:string" use="required"/>
        <xs:attribute name="bitrate" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="pixelFormat" type="xs:string" use="optional"/>          <!-- needed for choosing between DV variants -->
        <xs:attribute name="gopSize" type="xs:unsignedShort" use="optional"/>       <!-- set to zero for intra-only -->
        <xs:attribute name="maxBFrames" type="xs:unsignedShort" use="optional"/>    <!-- maximum number of B-frames between P-frames. zero disables B-frames -->
    </xs:complexType>

    <xs:complexType name="NLEJobAudioOutputType">
        <xs:sequence>
            <xs:element name="sequence" type="xs:int" maxOccurs="unbounded"/>       <!-- Audio sequences are mono, which means we need one sequence per output channel -->
            <xs:choice minOccurs="0" maxOccurs="1" >
                <xs:element name="channelLayout" type="xs:long"/>           <!-- binary representation, ex: 5.1 => 1551, see ffmpeg channel_layout.h -->

                <!--* name can be one or several of the following notations,
                    * separated by '+' or '|':
                    * - the name of an usual channel layout (mono, stereo, 4.0, quad, 5.0,
                    *   5.0(side), 5.1, 5.1(side), 7.1, 7.1(wide), downmix);
                    * - the name of a single channel (FL, FR, FC, LFE, BL, BR, FLC, FRC, BC,
                    *   SL, SR, TC, TFL, TFC, TFR, TBL, TBC, TBR, DL, DR);
                    * - a number of channels, in decimal, followed by 'c', yielding
                    *   the default channel layout for that number of channels (@see
                    *   av_get_default_channel_layout);
                    * - a channel layout mask, in hexadecimal starting with "0x" (see the
                    *   AV_CH_* macros).
                    *
                    * Example: "stereo+FC" = "2c+FC" = "2c+1c" = "0x7" git -->
                <xs:element name="channelLayoutName" type="xs:string"/>
            </xs:choice>
        </xs:sequence>
        <xs:attribute name="uri" type="xs:anyURI" use="optional"/>                  <!-- used when outputing OPAtom -->
        <xs:attribute name="codec" type="xs:string" use="required"/>
```

```
        <xs:attribute name="bitrate" type="xs:unsignedInt" use="optional"/>          <!-- not applicable to PCM -->
    </xs:complexType>

    <xs:complexType name="NLEJobOutputType">
        <xs:sequence>
            <xs:element name="video" type="tns:NLEJobVideoOutputType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="audio" type="tns:NLEJobAudioOutputType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="uri" type="xs:anyURI" use="optional"/>                   <!-- unique URIs are set in each <video> and <audio> element when outputing OPAtoms, hence optional here -->
        <xs:attribute name="containerFormat" type="xs:string" use="required"/>
        <xs:attribute name="umid" type="tns:UMIDType" use="optional"/>               <!-- Should be set when outputing OPAtom - generated otherwise -->

        <!-- MaterialPackage -> Name for MXF.
             Not applicable to most other formats (except maybe MOV).
        -->
        <xs:attribute name="clipName" type="xs:string" use="optional"/>
    </xs:complexType>

    <xs:complexType name="NLEJob2VideoOutputType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:ComplexJobVideoOutputType">
                <xs:sequence>
                    <xs:element name="uri" type="xs:anyURI" minOccurs="0" maxOccurs="1" />                <!-- used when outputing OPAtom -->
                    <xs:element name="sequence" type="xs:int" minOccurs="1" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="NLEJob2MixTypeInput">
        <xs:sequence>
            <xs:element name="effect" type="tns:EffectType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="sequence" type="xs:int" use="required"/>
        <xs:attribute name="gain" type="xs:float" use="required"/>
    </xs:complexType>

    <xs:complexType name="NLEJob2MixType">
        <xs:sequence>
            <xs:element name="input" type="tns:NLEJob2MixTypeInput" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="NLEJob2AudioOutputType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:ComplexJobAudioOutputType">
                <xs:sequence>
```

```
                    <xs:element name="uri" type="xs:anyURI" minOccurs="0" maxOccurs="1
↪" />                <!-- used when outputing OPAtom -->
                    <xs:element name="sequence" type="xs:int" maxOccurs="unbounded" />
                    <xs:element name="mix" type="tns:NLEJob2MixType" maxOccurs=
↪"unbounded" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="NLEJob2OutputType">
        <xs:sequence>
            <xs:element name="format" type="tns:ComplexJobOutputFormatType" minOccurs=
↪"1" maxOccurs="1"/>
            <xs:element name="video" type="tns:NLEJob2VideoOutputType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="audio" type="tns:NLEJob2AudioOutputType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

*NLEJobDocument*

```
    <xs:element name="NLEJobDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:NLEJobType"/>
    <xs:complexType name="NLEJobType">
        <xs:complexContent>
            <xs:extension base="tns:TranscoderJobType">
        <xs:sequence>
            <xs:element name="frameRate" type="tns:FrameRateType"/>
            <xs:element name="width" type="xs:unsignedShort"/>
            <xs:element name="height" type="xs:unsignedShort"/>
            <xs:element name="dar" type="tns:AspectRatioType"/>
            <xs:element name="sampleRate" type="xs:unsignedInt"/>

            <!-- What frame to pause the rendering at.
                 See ticket #106.
            -->
            <xs:element name="pauseFrame" type="xs:unsignedInt" minOccurs="0"/>

            <xs:element name="videoClip" type="tns:VideoClipType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="audioClip" type="tns:AudioClipType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="subtitleClip" type="tns:SubtitleClipType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="sequence" type="tns:NLEJobSequenceType" maxOccurs=
↪"unbounded"/>
            <xs:element name="output" type="tns:NLEJobOutputType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
            <xs:element name="output2" type="tns:NLEJob2OutputType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- QueueJob -->
```

*QueueJobDocument*

```
    <xs:element name="QueueJobDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:QueueJobType"/>
    <xs:complexType name="QueueJobType">
        <xs:sequence>
            <!-- Ordered list of jobs to run. Recursion (putting QueueJobs in job elements) is allowed -->
            <xs:element name="job" type="tns:JobRequestChoiceType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <!-- For specifying one of the valid job request types.
         Also useful since a virtual funtion in the Job class can return JobRequestChoiceType and have each implementation fill in the correct request -->
    <xs:complexType name="JobRequestChoiceType">
        <xs:choice>
            <xs:element name="timelineRequest" type="tns:TimelineJobRequestType"/>
            <xs:element name="complexRequest" type="tns:ComplexJobType"/>
            <xs:element name="movIndexRequest" type="tns:MOVIndexJobType"/>
            <xs:element name="mxfTimecodeExtractionRequest" type="tns:MXFTimecodeExtractionJobType"/>
            <xs:element name="mxfOp1bRequest" type="tns:MXFOp1bJobType"/>
            <xs:element name="segmentationRequest" type="tns:SegmentationJobType"/>
            <xs:element name="nleRequest" type="tns:NLEJobType"/>
            <xs:element name="queueRequest" type="tns:QueueJobType"/>
            <xs:element name="durationRequest" type="tns:DurationJobType"/>
            <xs:element name="shapeDeductionRequest" type="tns:ShapeDeductionJobType"/>
            <xs:element name="xmpRequest" type="tns:XMPJobType"/>
            <xs:element name="hashRequest" type="tns:HashJobType"/>
            <xs:element name="transferRequest" type="tns:TransferJobType"/>
            <xs:element name="aafGeneratorRequest" type="tns:AAFGeneratorJobType"/>
        </xs:choice>
    </xs:complexType>

    <xs:complexType name="JobLogEntryType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="timestamp" type="xs:dateTime"/>
                <xs:attribute name="level" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="SequenceRangeType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="start" type="xs:long" use="required"/>
                <xs:attribute name="width" type="xs:int" use="optional" />
                <xs:attribute name="count" type="xs:long" use="optional"/>
                <xs:attribute name="wildcard" type="xs:string" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="JobInputProgressType">
        <xs:sequence>
```

```
            <!-- Highest (timestamp + duration - startTime) of all packets processed for this input so far -->
            <xs:element name="mediaTime" type="tns:TimeCodeType"/>

            <!-- Duration of file, if known -->
            <xs:element name="duration" type="tns:TimeCodeType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <!-- For returning job status in various cases -->
```

*JobStatusDocument*

```
    <xs:element name="JobStatusDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:JobStatusType" />
    <xs:complexType name="JobStatusType">
        <xs:sequence>
            <xs:element name="statusUri" type="xs:anyURI"/>
    <!-- URI from to which a JobStatusRequestDocument can be sent to poll status -->
            <xs:element name="id" type="tns:SiteIdType"/>
            <xs:element name="isRunning" type="xs:boolean"/>
            <xs:element name="isPaused" type="xs:boolean"/>
            <xs:element name="walltime" type="xs:double"/>
            <xs:element name="exitcode" type="xs:int" minOccurs="0"/>
    <!-- Exit code if done running -->
            <xs:element name="message" type="xs:string" minOccurs="0"/>
    <!-- Possible exit message (exceptions, malformed requests etc.) -->
            <xs:element name="log" type="tns:JobLogEntryType"
                minOccurs="0" maxOccurs="unbounded"/>
    <!-- Log entries with timestamps and levels -->
            <xs:element name="request" type="tns:JobRequestChoiceType" minOccurs="0"/>
    <!-- the request that started this job -->
            <xs:element name="inputProgress" type="tns:JobInputProgressType"
                minOccurs="0" maxOccurs="unbounded"/>
    <!-- Amount of media processed per input file -->
            <xs:element name="progress" type="xs:float" minOccurs="0"/>
    <!-- Overvall percentage of media processed so far.

         It is only set if all inputs have known durations -->
            <xs:element name="estimatedTimeLeft" type="xs:float" minOccurs="0"/>
    <!-- walltime/(progress/100) - walltime -->
            <xs:element name="thumbnail" type="tns:ThumbnailInfoType"
                minOccurs="0" maxOccurs="unbounded" />
    <!-- Info on thumbnails generated -->
            <xs:element name="shapeDeductionResponse" type="tns:ShapeDeductionResponse" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="durationResponse" type="tns:DurationResponse" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="xmpResponse" type="tns:XMPResponse" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="hashResponse" type="tns:HashResponse" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="transferResponse" type="tns:TransferResponse" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="aafGeneratorResponse" type="tns:AAFGeneratorResponse" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
```

```
    <xs:complexType name="XMPResponse">
        <xs:sequence>
            <xs:element name="id" type="xs:string" />
            <xs:element name="uri" type="xs:anyURI" />
            <xs:element name="xmp" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="HashResponse">
        <xs:sequence>
            <xs:element name="id" type="xs:string" />
            <xs:element name="uri" type="xs:anyURI" />
            <xs:element name="function" type="xs:string" />
            <xs:choice>
                <xs:element name="hash" type="xs:string" />
                <xs:element name="error" type="xs:string" />
            </xs:choice>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="TransferResponse">
        <xs:sequence>
            <xs:element name="id" type="xs:string" />
            <xs:element name="sourceUri" type="xs:anyURI" minOccurs="1" maxOccurs=
↪"unbounded" />
            <xs:element name="destinationUri" type="xs:anyURI" minOccurs="1"␣
↪maxOccurs="unbounded" />
            <xs:element name="error" type="xs:string" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ShapeDeductionResponse">
        <xs:sequence>
            <xs:element name="id" type="xs:string" />
            <xs:element name="uri" type="xs:anyURI" minOccurs="0" maxOccurs="1"/>
            <xs:element name="range" type="tns:SequenceRangeType" maxOccurs="unbounded
↪"/>
            <xs:element name="shape" type="tns:ShapeType" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="DurationResponse">
        <xs:sequence>
            <xs:element name="id" type="xs:string" />
            <xs:element name="uri" type="xs:anyURI" />
            <xs:element name="duration" type="tns:DurationType" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AAFGeneratorResponse">
        <xs:sequence>
            <xs:choice>
                <xs:element name="aaf" type="xs:hexBinary" />
                <xs:element name="xml" type="xs:string" />
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
```

```
    <xs:complexType name="ThumbnailInfoType">
        <xs:sequence>
            <xs:element name="timeCode" type="tns:TimeCodeType" />
            <xs:element name="uri" type="xs:string" />
        </xs:sequence>
    </xs:complexType>
```

*JobStatusListDocument*

```
    <xs:element name="JobStatusListDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:JobStatusListType" />

    <xs:complexType name="JobStatusListType">
        <xs:sequence>
            <xs:element name="hits" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="job" type="tns:JobStatusType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Deprecated -->
```

*SimpleTimelineDocument*

```
    <xs:element name="SimpleTimelineDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:SimpleTimelineType" />
    <xs:complexType name="SimpleTimelineType">
        <xs:sequence>
            <xs:element name="destinationURI" type="xs:string"/>
            <xs:element name="source" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="sequence" type="xs:int"/>             <!-- For guaranteeing ordering -->
                        <xs:choice>                                             <!-- Use URI when talking to transcoder -->
                            <xs:element name="uri" type="xs:anyURI"/>
                            <xs:element name="siteId" type="tns:SiteIdType"/>
                        </xs:choice>
                        <xs:element name="interval"
                                    type="tns:TimeIntervalType"/>               <!-- Interval to use -->
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```

*TimelineDocument*

```
    <xs:element name="TimelineDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:TimelineType" />
    <xs:complexType name="TimelineType">
        <xs:sequence>
            <xs:element name="destination" type="tns:ShapeType"/>
            <xs:element name="track">
                <xs:complexType>
```

```
                    <xs:sequence>
                        <xs:element name="source" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="sequence" type="xs:int"/>
↪       <!-- For guaranteeing ordering -->
                                    <xs:choice>
↪       <!-- Use URI when talking to transcoder -->
                                        <xs:element name="uri" type="xs:anyURI"/>
                                        <xs:element name="siteId" type="tns:SiteIdType
↪"/>
                                    </xs:choice>
                                    <xs:element name="track" type="xs:string"/>
↪       <!-- Which track in the source to use for this segment -->
                                    <xs:element name="interval"
                                        type="tns:TimeIntervalType"/>
↪   <!-- Interval to use -->
                                    <xs:element name="transition"
                                        type="tns:TimeCodeType" minOccurs="0"/>
↪   <!-- Length of transition period following the interval, if any -->
                                    <xs:element name="effect"
                                        type="xs:string" minOccurs="0"/>
↪   <!-- Transition effect to use. Default if not set -->

                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                    <xs:attribute name="index" type="xs:string"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <!-- Types used for indexing files on tape and other media where seeking is very␣
↪expensive -->
    <xs:complexType name="PartialFileRandomIndexType">
        <xs:sequence>
            <xs:element name="packet" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="pts" type="xs:long"/>             <!-- In␣
↪MediaComponentType/timeBase units -->
                    <xs:attribute name="dts" type="xs:long"/>             <!-- In␣
↪MediaComponentType/timeBase units -->
                    <xs:attribute name="offset" type="xs:unsignedLong"/>
                    <xs:attribute name="length" type="xs:unsignedInt"/>
                    <xs:attribute name="duration" type="xs:unsignedInt"/> <!-- In␣
↪MediaComponentType/timeBase units -->
                    <xs:attribute name="stream" type="xs:unsignedByte"/>  <!--␣
↪References MediaComponentType/essenceStreamId -->
                    <xs:attribute name="isKeyFrame" type="xs:boolean"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="PartialFileDVDescriptorType">
        <xs:sequence>
```

```
            <xs:element name="frameSize" type="xs:unsignedInt"/>
            <xs:element name="frameCount" type="xs:unsignedInt"/>
        </xs:sequence>
    </xs:complexType>
```

*PartialFileDescriptorDocument*

```
    <xs:element name="PartialFileDescriptorDocument" type="tns:PartialFileDescriptorType"/>
    <xs:complexType name="PartialFileDescriptorType">
        <xs:sequence>
            <xs:element name="label" type="xs:string" minOccurs="0"/>

            <xs:element name="transcoderVersion" type="xs:string" minOccurs="0"/>

            <!-- Corresponds to StartTimecode in TimecodeComponent in MXF -->
            <xs:element name="startTimecode" type="xs:long" minOccurs="0"/>

            <!-- Corresponds to RoundedTimeBase in TimecodeComponent in MXF -->
            <xs:element name="roundedTimeBase" type="xs:int" minOccurs="0"/>

            <!-- Corresponds to DropFrame in TimecodeComponent in MXF -->
            <xs:element name="dropFrame" type="xs:boolean" minOccurs="0"/>

            <xs:element name="containerComponent" type="tns:ContainerComponentType" minOccurs="0"/>
            <xs:element name="audioStream" type="tns:AudioComponentType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="videoStream" type="tns:VideoComponentType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:choice>
                <xs:element name="dvDescriptor" type="tns:PartialFileDVDescriptorType"/>    <!-- Used for DV -->
                <xs:element name="index" type="tns:PartialFileRandomIndexType"/>      <!-- Used for MXF, AVI and similar -->
            </xs:choice>
        </xs:sequence>
    </xs:complexType>

    <!-- Types for requesting a byte range for a time interval in a PartialFileDescriptorDocument -->
```

*ByteRangeRequestDocument*

```
    <xs:element name="ByteRangeRequestDocument" type="tns:ByteRangeRequestType"/>
    <xs:complexType name="ByteRangeRequestType">
        <xs:sequence>
            <xs:element name="interval" type="tns:TimeIntervalType"/>
            <xs:element name="descriptor" type="tns:PartialFileDescriptorType"/>
        </xs:sequence>
    </xs:complexType>
```

*ByteRangeResponseDocument*

```
    <xs:element name="ByteRangeResponseDocument" type="tns:ByteRangeResponseType"/>
    <xs:complexType name="ByteRangeResponseType">
        <xs:sequence>
            <xs:element name="start" type="xs:unsignedLong"/>
```

```
            <xs:element name="end" type="xs:unsignedLong"/>
        </xs:sequence>
    </xs:complexType>

    <!-- XML types for dealing with DMS-1 metadata in MXF -->
    <xs:simpleType name="InstanceUID">
        <xs:restriction base="xs:hexBinary"/>
    </xs:simpleType>
    <xs:simpleType name="ULType">   <!-- "UL" seems a bit short, so I'm picking ULType -->
        <xs:restriction base="xs:hexBinary"/>
    </xs:simpleType>
    <xs:complexType name="MDObjectBase">
        <xs:attribute name="name" type="xs:string" use="optional"/>
        <xs:attribute name="ul" type="tns:ULType" use="required"/>
    </xs:complexType>
    <xs:complexType name="MDObjectWeakReference">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:MDObjectBase">
                <xs:sequence>
                    <xs:element name="target" type="tns:InstanceUID"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="MDObjectLeaf">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:MDObjectBase">
                <xs:sequence>
                    <xs:choice>
                        <xs:element name="hexValue" type="xs:hexBinary"/>
                        <xs:element name="stringValue" type="xs:string"/>
                    </xs:choice>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="MDObjectNode">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:MDObjectBase">
                <xs:sequence>
                    <xs:element name="leaf" type="tns:MDObjectLeaf" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="child" type="tns:MDObjectNode" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="strongReference" type="tns:MDObjectStrongReference" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="weakReference" type="tns:MDObjectWeakReference" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="instanceUid" type="tns:InstanceUID" use="optional"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

```
    <xs:complexType name="MDObjectStrongReference">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:MDObjectNode">
                <xs:attribute name="referenceUl" type="tns:ULType" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="MDSegment">
        <xs:sequence>
            <xs:element name="interval" type="tns:TimeIntervalType"/>
            <xs:element name="dms1Framework" type="tns:MDObjectNode"/>  <!-- DMS-1 scene or clip framework -->
        </xs:sequence>
    </xs:complexType>
```

*DMS1Document*

```
    <xs:element name="DMS1Document" type="tns:DMS1Type"/>
    <xs:complexType name="DMS1Type">
        <xs:sequence>
            <xs:element name="partition" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="materialPackage" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="staticTrack" minOccurs="0" maxOccurs="unbounded">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element name="dms1Framework" type="tns:MDObjectNode"/>  <!-- DMS-1 production framework -->
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
                                    <xs:element name="eventTrack" minOccurs="0" maxOccurs="unbounded">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element name="segment" type="tns:MDSegment" minOccurs="0" maxOccurs="unbounded"/>
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                    <xs:attribute name="offset" type="xs:long"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="UMIDType">
```

```
        <xs:restriction base="xs:hexBinary">
            <!-- UMIDs are 256 bits -->
            <xs:minLength value="32"/>
            <xs:maxLength value="32"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="MXFTimestampType">
        <xs:sequence>
            <!-- Corresponds to mxfTimestamp in libMXF -->
            <xs:element name="year" type="xs:short"/>
            <xs:element name="month" type="xs:unsignedByte"/>
            <xs:element name="day" type="xs:unsignedByte"/>
            <xs:element name="hour" type="xs:unsignedByte"/>
            <xs:element name="min" type="xs:unsignedByte"/>
            <xs:element name="sec" type="xs:unsignedByte"/>
            <xs:element name="qmsec" type="xs:unsignedByte"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="PackageTrackType">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>

            <!-- Physical track ID, like audio channels -->
            <xs:element name="number" type="xs:int"/>

            <xs:element name="isPicture" type="xs:boolean"/>

            <xs:element name="is50FPS" minOccurs="0" maxOccurs="1" type="xs:boolean"/>

            <xs:element name="frameRate" minOccurs="0" maxOccurs="1" type="tns:
↪FrameRateType" />

            <!-- Length of track in edit units -->
            <xs:element name="length" type="xs:int"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="MaterialPackageTrackType">
        <xs:complexContent>
            <xs:extension base="tns:PackageTrackType">
                <xs:sequence>
                    <xs:element name="sourcePackageID" type="tns:UMIDType"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TapePackageTrackType">
        <xs:complexContent>
            <xs:extension base="tns:PackageTrackType">
                <xs:sequence>
                    <xs:element name="trackID" type="xs:int"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

```
    <xs:complexType name="PackageType">
        <xs:sequence>
            <xs:element name="umid" type="tns:UMIDType"/>
            <xs:element name="timestamp" type="tns:MXFTimestampType"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="MaterialPackageType">
        <xs:complexContent>
            <xs:extension base="tns:PackageType">
                <xs:sequence>
                    <xs:element name="track" type="tns:MaterialPackageTrackType" minOccurs="1" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TapePackageType">
        <xs:complexContent>
            <xs:extension base="tns:PackageType">
                <xs:sequence>
                    <xs:element name="track" type="tns:TapePackageTrackType" minOccurs="1" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Needed for muxing OpAtom -->
    <xs:complexType name="MXFPackagesType">
        <xs:sequence>
            <xs:element name="materialPackage" type="tns:MaterialPackageType"/>
            <xs:element name="tapePackage" type="tns:TapePackageType" minOccurs="0"/>

            <!-- Material package track to link to file package track, like "V1" -->
            <xs:element name="materialTrackName" type="xs:string"/>

            <!-- Tape package track to link file package track to, like "V1".
                 Must be set if tapePackage is set.
            -->
            <xs:element name="tapeTrackName" type="xs:string" minOccurs="0"/>
            <xs:element name="projectEditRate" type="tns:FrameRateType" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
    </xs:complexType>

    <!--- START METADATA TYPES -->

    <xs:complexType name="KeyValuePairType">
        <xs:sequence>
            <xs:element name="key" minOccurs="1" maxOccurs="1" type="xs:string"/>
            <xs:element name="value" minOccurs="1" maxOccurs="1" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <!--
```

```
        MetadataReferenceType is only used when posting new metadata.
        They only need to be unique within the same MetadataDocument.
        The middleware will transform them to proper UUIDs, unless they already formatted as UUIDs pointing to existing metadata.
    -->
   <xs:simpleType name="MetadataReferenceType">
       <xs:restriction base="xs:string"/>
   </xs:simpleType>

   <xs:complexType name="MetadataGroupValueType">
       <xs:sequence>
           <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
           <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
           <xs:element name="referenced" type="tns:MetadataReferencedType" minOccurs="0" maxOccurs="1"/>
           <xs:choice>
               <xs:sequence>
                   <xs:element name="field" type="tns:MetadataFieldValueType" minOccurs="0" maxOccurs="unbounded"/>
                   <xs:element name="group" type="tns:MetadataGroupValueType" minOccurs="0" maxOccurs="unbounded"/>
               </xs:sequence>
               <xs:sequence>
                   <xs:element name="reference" type="tns:MetadataReferenceType" minOccurs="1" maxOccurs="1"/>
               </xs:sequence>
           </xs:choice>
           <xs:element name="data" minOccurs="0" maxOccurs="unbounded" type="tns:KeyValuePairType"/>
       </xs:sequence>
       <xs:attributeGroup ref="tns:MetadataValueAttributes"/>
       <xs:attribute name="inheritance" type="xs:string" use="optional"/>
       <xs:attribute name="cycle" type="xs:boolean" use="optional"/>
   </xs:complexType>

   <xs:complexType name="MetadataFieldValueType">
       <xs:sequence>
           <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
           <xs:element name="id" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
           <xs:element name="referenced" type="tns:MetadataReferencedType" minOccurs="0" maxOccurs="1"/>
           <xs:choice>
               <xs:element name="value" type="tns:MetadataValueType" minOccurs="0" maxOccurs="unbounded"/>
               <xs:element name="reference" type="tns:MetadataReferenceType" minOccurs="1" maxOccurs="1"/>
           </xs:choice>
           <xs:element name="data" minOccurs="0" maxOccurs="unbounded" type="tns:KeyValuePairType"/>
           <xs:element name="type" minOccurs="0" maxOccurs="1" type="tns:MetadataFieldType"/>
       </xs:sequence>
       <xs:attributeGroup ref="tns:MetadataValueAttributes"/>
       <xs:attribute name="track" type="xs:string" use="optional"/>
       <xs:attribute name="inheritance" type="xs:string" use="optional"/>
   </xs:complexType>
```

```
    <xs:complexType name="MetadataReferencedType">
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xs:attribute name="uuid" type="xs:string" use="required"/>
        <xs:attribute name="type" type="xs:string" use="required"/>
    </xs:complexType>

    <xs:complexType name="MetadataValueType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attributeGroup ref="tns:MetadataValueAttributes"/>
                <xs:attribute name="lang" type="xs:language" use="optional"/>
                <xs:attribute name="id" type="xs:string" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:attributeGroup name="MetadataValueAttributes">
        <xs:attribute name="uuid" type="tns:MetadataReferenceType" use="optional"/>
        <xs:attribute name="user" type="xs:string" use="optional"/>
        <xs:attribute name="timestamp" type="xs:dateTime" use="optional"/>
        <xs:attribute name="change" type="tns:SiteIdType" use="optional"/>
        <xs:attribute name="conflict" type="xs:boolean" use="optional"/>
        <xs:attribute name="mode"  type="tns:MetadataModeType" use="optional"/>
    </xs:attributeGroup>

    <xs:simpleType name="MetadataModeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="add"/>
            <xs:enumeration value="remove"/>
        </xs:restriction>
    </xs:simpleType>
```

*MetadataDocument*

```
    <xs:element name="MetadataDocument" xmlns:tns="http://xml.vidispine.com/schema/
↪vidispine" type="tns:MetadataType" />
    <xs:complexType name="MetadataType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="revision" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="template" type="xs:string" minOccurs="0" maxOccurs="1"/
↪ <!-- obsolete -->
            <xs:element name="group" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
            <xs:element name="timespan" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="field" type="tns:MetadataFieldValueType"␣
↪minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="group" type="tns:MetadataGroupValueType"␣
↪minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                    <xs:attribute name="start" type="xs:string" />
                    <xs:attribute name="end" type="xs:string" />
                    <xs:attribute name="base" type="xs:string" />
                </xs:complexType>
            </xs:element>
        </xs:sequence>
```

```
    </xs:complexType>

    <!-- END METADATA TYPES -->

    <!-- START METADATA FIELD TYPE TYPES -->

    <xs:simpleType name="MetadataFieldTypeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="date"/>
            <xs:enumeration value="date-noindex"/>
            <xs:enumeration value="date-sortable"/>
            <xs:enumeration value="float"/>
            <xs:enumeration value="float-noindex"/>
            <xs:enumeration value="float-sortable"/>
            <xs:enumeration value="integer"/>
            <xs:enumeration value="integer-noindex"/>
            <xs:enumeration value="integer-sortable"/>
            <xs:enumeration value="long"/>
            <xs:enumeration value="long-noindex"/>
            <xs:enumeration value="string"/>
            <xs:enumeration value="string-sortable"/>
            <xs:enumeration value="string-exact"/>
            <xs:enumeration value="string-exact-sortable"/>
            <xs:enumeration value="string-noindex"/>
            <xs:enumeration value="boolean"/>
            <xs:enumeration value="boolean-noindex"/>
            <xs:enumeration value="timeCode"/>
            <xs:enumeration value="timeCode-noindex"/>

        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="MetadataFieldIndexType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="noindex"/>
            <xs:enumeration value="index"/>
            <xs:enumeration value="extend"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="MetadataFieldFloatType">
        <xs:sequence>
            <xs:element name="minInclusive" type="xs:double" minOccurs="0" maxOccurs=
↪"1"/>
            <xs:element name="maxInclusive" type="xs:double" minOccurs="0" maxOccurs=
↪"1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="MetadataFieldIntegerType">
        <xs:sequence>
            <xs:element name="minInclusive" type="xs:int" minOccurs="0" maxOccurs="1"/
↪>
            <xs:element name="maxInclusive" type="xs:int" minOccurs="0" maxOccurs="1"/
↪>
        </xs:sequence>
    </xs:complexType>

```

```
    <xs:complexType name="MetadataFieldLongType">
        <xs:sequence>
            <xs:element name="minInclusive" type="xs:long" minOccurs="0" maxOccurs="1
↪"/>
            <xs:element name="maxInclusive" type="xs:long" minOccurs="0" maxOccurs="1
↪"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="MetadataFieldStringType">
        <xs:sequence>
            <xs:element name="minLength" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="maxLength" type="xs:int" minOccurs="0" maxOccurs="1"/>
            <xs:element name="pattern" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

*MetadataFieldDocument*

```
    <xs:element name="MetadataFieldDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:MetadataFieldType" />
    <xs:complexType name="MetadataFieldType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="schema" type="tns:MetadataSchemaElementType" minOccurs=
↪"0" maxOccurs="1"/>
            <xs:element name="type" type="tns:MetadataFieldTypeType" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="index" type="tns:MetadataFieldIndexType" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="fullText" type="xs:boolean" minOccurs="0" maxOccurs="1"/
↪>
            <xs:element name="caseSensitiveSorting" type="xs:boolean" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="constraint" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="dataset" type="xs:string" minOccurs="1"␣
↪maxOccurs="1"/>
                        <xs:element name="levelProperty" type="xs:string" minOccurs="1
↪" maxOccurs="1"/>
                        <xs:element name="levelValue" type="xs:string" minOccurs="1"␣
↪maxOccurs="1"/>
                        <xs:element name="value" type="xs:string" minOccurs="1"␣
↪maxOccurs="1"/>
                        <xs:choice>
                            <xs:element name="validationGroup" type="xs:string"␣
↪minOccurs="0" maxOccurs="1"/>
                            <xs:element name="parent" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
                        </xs:choice>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:choice minOccurs="0" maxOccurs="1">
                <xs:element name="floatRestriction" type="tns:MetadataFieldFloatType"/
↪>
                <xs:element name="integerRestriction" type="tns:
↪MetadataFieldIntegerType"/>
```

```
                <xs:element name="longRestriction" type="tns:MetadataFieldLongType"/>
                <xs:element name="stringRestriction" type="tns:MetadataFieldStringType
↪"/>
            </xs:choice>
            <xs:element name="data" minOccurs="0" maxOccurs="unbounded" type="tns:
↪KeyValuePairType"/>
            <xs:element name="values" minOccurs="0" maxOccurs="1" type="tns:
↪SimpleMetadataType"/>
            <xs:element name="defaultValue" type="xs:string" minOccurs="0" maxOccurs=
↪"1"/>
            <xs:element name="externalId" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded" />
            <xs:element name="origin" type="xs:string" minOccurs="0" maxOccurs="1" />
            <xs:element name="boost" type="xs:float" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
        <xs:attribute name="system" type="xs:string" use="optional"/>
        <xs:attribute name="sortable" type="xs:boolean" use="optional"/>
        <xs:attribute name="inheritance" type="xs:string" use="optional"/>
    </xs:complexType>

    <!-- END METADATA FIELD TYPE TYPES -->

    <xs:complexType name="SimpleMetadataType">
        <xs:sequence>
            <!-- TODO: use tns:KeyValuePairType instead -->
            <xs:element name="field" minOccurs="0" maxOccurs="unbounded" >
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="key" type="xs:string" minOccurs="1"␣
↪maxOccurs="1" />
                        <xs:element name="value" type="xs:string" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <!-- Decode/encode permissions and license document. Wildcards are allowed.
         Example how a license document allowing any input to be transcoded to H.
↪264+MP3 might look like:
             <TranscoderLicenseStatusDocument>
                 <mayDecode>*</mayDecode>
                 <mayEncode>*mp3*</mayEncode>
                 <mayEncode>*264*</mayEncode>
             </TranscoderLicenseStatusDocument>
    -->
```

*TranscoderLicenseStatusDocument*

```
    <xs:element name="TranscoderLicenseStatusDocument" xmlns:tns="http://xml.
↪vidispine.com/schema/vidispine" type="tns:TranscoderLicenseStatusType"/>
    <xs:complexType name="TranscoderLicenseStatusType">
        <xs:sequence>
            <xs:element name="mayDecode" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
            <xs:element name="mayEncode" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
            <xs:element name="production" type="xs:boolean" minOccurs="0" maxOccurs="1
↪"/>
```

```
        </xs:sequence>
    </xs:complexType>

    <!-- Returned by the transcoder's duration resource -->
```

*DurationDocument*

```
    <xs:element name="DurationDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:DurationType"/>
    <xs:complexType name="DurationType">
        <xs:sequence>
            <!-- duration = max_x{ptsInterval.end_x} - min_x{ptsInterval.start_x} -->
            <xs:element name="duration" type="tns:TimeCodeType"/>

            <!-- Information about the individual video streams
                 duration = stream.end - stream.start
            -->
            <xs:element name="stream" type="tns:StreamIntervalType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="StreamIntervalType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:TimeIntervalType">
                <!-- AKA essenceStreamId -->
                <xs:attribute name="index" type="xs:unsignedShort" use="required"/>

                <!-- number of frames decoded -->
                <xs:attribute name="numberOfFrames" type="xs:int" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

*TranscoderVersionDocument*

```
    <xs:element name="TranscoderVersionDocument" xmlns:tns="http://xml.vidispine.com/schema/vidispine" type="tns:TranscoderVersionType"/>
    <xs:complexType name="TranscoderVersionType">
        <xs:sequence>
            <xs:element name="version" type="xs:string" minOccurs="1" maxOccurs="1" />
            <xs:element name="submodule" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="version" type="xs:string" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="info" type="tns:KeyValuePairType" minOccurs="0" maxOccurs="unbounded" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="feature" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
```

```
                        <xs:element name="version" type="xs:string" minOccurs="0"
↪maxOccurs="1"/>
                        <xs:element name="info" type="tns:KeyValuePairType" minOccurs=
↪"0" maxOccurs="unbounded" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>


</xs:schema>
```

### 17.38.3 transcoder.xsd

Common elements to API and Transcoder.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.vidispine.com/schema/vidispine"
    elementFormDefault="qualified"
    xmlns:tns="http://xml.vidispine.com/schema/vidispine">
    <!-- Schemas for transcoder configurations -->
    <xs:simpleType name="GUIDType">
        <xs:restriction base="xs:string">
            <xs:pattern value="(\{([0-9a-fA-F]){8}-([0-9a-fA-F]){4}-([0-9a-fA-F]){4}-
↪([0-9a-fA-F]){4}-([0-9a-fA-F]){12}\})"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="CarbonPreset">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="description" type="xs:string"/>
            <xs:element name="containerFormat" type="xs:string"/>
            <xs:element name="videoCodec" type="xs:string"/>
            <xs:element name="audioCodec" type="xs:string"/>
            <xs:element name="displayAspectRatio" type="tns:AspectRatioType"/>
            <xs:element name="GUID" type="tns:GUIDType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="PluginType">
        <xs:sequence>
            <xs:element name="alias" type="xs:string"/>
            <xs:element name="fileName" type="xs:string"/>  <!-- Name of .dll/.so
↪file -->
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="AddressPortType">
        <xs:sequence>
            <xs:choice>
                <xs:element name="host" type="xs:string"/>
                <xs:element name="address" type="xs:string"/>
            </xs:choice>
            <xs:element name="port" type="xs:int"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="StatsDRecieverType">
```

```
        <xs:sequence>
            <xs:element name="destination" type="tns:AddressPortType"/>
            <xs:element name="prefix" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="REDDecoderType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="CPU"/>
            <xs:enumeration value="CUDA"/> <!-- default, will fall back to CPU if CUDA is not available -->
            <xs:enumeration value="OPENCL"/>
        </xs:restriction>
    </xs:simpleType>
```

*TranscoderConfigurationDocument*

```
    <xs:element name="TranscoderConfigurationDocument" type="tns:TranscoderConfigurationType" />
    <xs:complexType name="TranscoderConfigurationType">
            <xs:sequence>
                <!-- TODO: replace these two with an AddressPortType element -->
                <xs:element name="address" type="xs:string"/>
                <xs:element name="port" type="xs:int"/>

                <!-- Number of threads to use in encoders.
                     Applies mostly to MainConcept and libx264 as of writing.

                      Not set = Leave thread count alone. This means a single thread for libavcodec, auto for MainConcept.
                      0       = Auto.
                      >=1     = Use specified number of threads. Should be somewhere around 150% of the number of cores.
                -->
                <xs:element name="encoderThreads" type="xs:int" minOccurs="0"/>
                <xs:element name="decoderOfferThreads" type="xs:int" minOccurs="0"/>
                <xs:element name="apiUsername" type="xs:string"/>
                <xs:element name="apiPassword" type="xs:string"/>

                <!-- If set, grab TranscoderLicenseStatusDocuments from apiURL/API/transcoder-validate
                     Else, wait for such documents to be PUT on /license
                     See T#114
                -->
                <xs:element name="apiURL" type="tns:AddressPortType" minOccurs="0"/>
                <xs:element name="thumbnailResolution" type="tns:ResolutionType"/>
                <xs:element name="thumbnailPeriod" type="tns:TimeCodeType" minOccurs="0" />
                <xs:element name="bilinearEffects" type="xs:boolean"/>              <!-- If true, use bilinear filtering for effects -->
                <xs:element name="carbonServer" minOccurs="0" maxOccurs="unbounded" type="tns:AddressPortType"/>
                <xs:element name="carbonPreset" type="tns:CarbonPreset" minOccurs="0" maxOccurs="unbounded"/>
                                    <xs:element name="faceDetectorPlugin" type="tns:PluginType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="dataPath" type="xs:string"/>
                <xs:element name="presetPath" type="xs:string"/>
                <xs:element name="tempPath" type="xs:string" minOccurs="0" />
```

```
84                  <xs:element name="proresDecoder" minOccurs="0" type="tns:
↪AddressPortType"/>
85                  <xs:element name="proresEncoder" minOccurs="0" type="tns:
↪AddressPortType"/>
86                  <xs:element name="vp6Encoder" minOccurs="0" type="tns:AddressPortType
↪"/>
87                  <xs:element name="vp6EncoderPoolSize" minOccurs="0" type="xs:int"/>
88                  <xs:element name="logo" minOccurs="0" type="xs:string"/>
89                  <xs:element name="imagemagick" minOccurs="0" maxOccurs="unbounded"␣
↪type="tns:KeyValuePairType"/>
90                  <xs:element name="logLevel" minOccurs="0" type="xs:string"/>
91                  <xs:element name="statsd" type="tns:StatsDRecieverType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
92                  <xs:element name="readBufferLength" type="xs:int" minOccurs="0"/>
93                  <xs:element name="dataBufferSize" type="xs:int" minOccurs="0"/>
94                  <xs:element name="dataBufferWriteSize" type="xs:int" minOccurs="0"/>
95                  <xs:element name="dataBufferFlushTime" type="xs:int" minOccurs="0"/>
96                  <xs:element name="colorProfilePath" type="xs:string" minOccurs="0"␣
↪maxOccurs="unbounded"/>
97                  <xs:element name="redDecoderType" type="tns:REDDecoderType" minOccurs=
↪"0" maxOccurs="1"/>
98              </xs:sequence>
99      </xs:complexType>
100
101     <!-- Schemas for communicating with Carbon -->
102     <xs:complexType name="CarbonJobInfoType">
103         <xs:sequence>
104             <xs:element name="Failures" minOccurs="0">
105                 <xs:complexType>
106                     <xs:sequence>
107                         <xs:element name="Warnings" type="xs:string" minOccurs="0"/>
108                         <xs:element name="Errors" type="xs:string" minOccurs="0"/>
109                     </xs:sequence>
110                 </xs:complexType>
111             </xs:element>
112         </xs:sequence>
113         <xs:attribute name="Name" type="xs:string" use="required"/>
114         <xs:attribute name="GUID" type="tns:GUIDType" use="required"/>
115         <xs:attribute name="State" type="xs:string" use="required"/>
116         <xs:attribute name="Status" type="xs:string" use="required"/>
117         <xs:attribute name="Progress.DWD" type="xs:int" use="required"/>
118         <xs:attribute name="Speed.DBL" type="xs:double" use="optional"/>
↪   <!-- Not documented -->
119         <xs:attribute name="Description" type="xs:string" use="required"/>
120         <xs:attribute name="User" type="xs:string" use="required"/>
121         <xs:attribute name="SourceDescription" type="xs:string" use="required"/>
122         <xs:attribute name="AgentIP" type="xs:string" use="required"/>
123         <xs:attribute name="Priority.DWD" type="xs:int" use="required"/>
124         <xs:attribute name="Capabilities.DWD" type="xs:int" use="optional"/>
↪   <!-- Not documented -->
125         <xs:attribute name="DeleteProcessedSource.DWD" type="xs:int" use="optional"/>␣
↪   <!-- Not documented -->
126         <xs:attribute name="DeleteRealAsset.DWD" type="xs:int" use="optional"/>
↪   <!-- Not documented -->
127         <xs:attribute name="CheckInTime" type="xs:string" use="required"/>
128         <xs:attribute name="CheckInTime_CNLT" type="xs:string" use="required"/>
129         <xs:attribute name="CheckInTime_SCM" type="xs:string" use="required"/>
130         <xs:attribute name="CheckInTimePrecise.QWD" type="xs:long" use="optional"/>
↪   <!-- Not documented -->
```

```
        <xs:attribute name="StartTime" type="xs:string" use="optional"/>
        <xs:attribute name="StartTime_CNLT" type="xs:string" use="optional"/>
        <xs:attribute name="StartTime_SCM" type="xs:string" use="optional"/>
        <xs:attribute name="CompletedTime" type="xs:string" use="optional"/>
        <xs:attribute name="CompletedTime_CNLT" type="xs:string" use="optional"/>
        <xs:attribute name="CompletedTime_SCM" type="xs:string" use="optional"/>
        <xs:attribute name="Error" type="xs:string" use="optional"/>
    <!-- Not documented -->
    </xs:complexType>
```

*Reply*

```
    <xs:element name="Reply">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="JobInfo" type="tns:CarbonJobInfoType" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="GUID" type="tns:GUIDType" use="optional"/>
            <xs:attribute name="Success" type="xs:string" use="required"/>
            <xs:attribute name="Error" type="xs:string" use="optional"/>
            <xs:attribute name="NrOfJobs.DWD" type="xs:int" use="optional"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

CHAPTER

EIGHTEEN

# RELEASE NOTES

This page contains the release notes for each Vidispine release. For older versions, see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

## 18.1 Prerequisites

The following prerequisite software is supported in this version.

### Operating systems

Vidispine server (VS):

- Ubuntu 18.04 LTS (64 bit)
- Ubuntu 20.04 LTS (64 bit)
- Red Hat Linux 7.0

Vidispine server agent (VSA):

- Ubuntu 18.04 LTS (64 bit) [1]
- Ubuntu 20.04 LTS (64 bit) [1]

### Databases

- PostgreSQL 9.6 and 11
- MySQL 5.6 and 5.7 [1]
- Microsoft SQL Server 2017 [1]

### Java

- Java 11

### Other

- Solr 8.1.0 (if installed manually)
- Zookeeper 3.4.6 (for HA systems) [1]
- Elasticsearch 6.8.1
- ActiveMQ 5.15

[1] Vidispine Enterprise edition only.

Changed in version 5.4.

- Support for Ubuntu 20.04 was added.

Changed in version 5.2.

- Support for CentOS 8 was added.

Changed in version 5.0.

- Support for Java 11 was added.
- Support for Java 8 was removed.
- Support for Solr 8.1.0 was added.
- Support for Solr 4.10.0 was removed.
- Support for Elasticsearch 6.8.1 was added.
- Support for Elasticsearch 5.6.x, 6.0.x, 6.1.x and 6.2.x was removed.
- Support for CentOS 6 was removed.
- Support for Ubuntu 16.04 was removed.

Changed in version 4.17.

- Support for PostgreSQL 11 and Ubuntu 18.04 was added.
- Support for PostgreSQL 9.1, PostgreSQL 9.3, MySQL 5.5 and Ubuntu 14.04 was removed.

## 18.2 Upgrade notes

### 18.2.1 General

- Your system must be running Vidispine version 4.0.x or later in order to upgrade to Vidispine 4.x.
- You may upgrade directly from release 4.0.x or later to e.g. 4.5, without first installing each intermediate release.
- When upgrading a major or minor (4.x to 4.y) version you should always *migrate the database*, for a maintenance release (4.13.x to 4.13.y) it is typically not needed.
- To be able to upgrade to version 5.0 of Vidispine you must first upgrade to Vidispine 4.17 and complete the database migration process.

### 18.2.2 Upgrading from 5.4 to 5.5

- APIinit is needed due to a new job step for analyze jobs.
- Solr: No changes to the documents. Re-indexing is not required.
- This release contains database migrations for both metadata and collections that may take time on larger systems. Please plan upgrades accordingly.

#### Breaking changes

- Searching with cursors: Previously, when reaching the end of a result set, the *next* cursor returned was null. With 5.5 a valid cursor is returned instead. This enables tailing searches (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#tailing-a-cursor). To check for end of search, either check hits being less than the requested number, or if the returned cursor is the same as the previous cursor.

- When enabling *indexCollectionItemOrder*, the order is in ascending order by the first entry of the item in the respective collection. (It is still recommended that this property is set to `false` for performance reasons.)

### 18.2.3 Upgrading from 5.3 to 5.4

- Solr: No changes to the documents. Re-indexing is not required.

### 18.2.4 Upgrading from 5.2 to 5.3

#### External identifier migration

*External identifiers* in VidiCore should be unique among the same entity type. But if there are multiple "overlapping" external identifier namespaces defined, duplicate identifiers could exist.

This has been fixed in 5.3 (#4350). A stronger constraint has been added to the table to prevent duplicates. However, if there are already duplicate entries in the table, manual intervention is required.

Below are the steps:

1. Use this SQL statement to identify duplicated entries:

```
SELECT i2.c_entity_type, i2.c_entity_id, i2.c_identifier_value, i2.c_
↪identifier_name
FROM
(SELECT c_entity_type, c_identifier_value FROM t_externalidentry
    GROUP BY c_entity_type, c_identifier_value HAVING COUNT(*) > 1) i1
INNER JOIN
(SELECT * from t_externalidentry ) i2
ON i2.c_entity_type = i1.c_entity_type AND i2.c_identifier_value = i1.c_
↪identifier_value
ORDER BY c_identifier_value;
```

2. If the above query doesn't return anything, meaning no duplicates, you can skip the rest and processed with automatic database migration.

3. If there are duplicate entries, you need to decide which one(s) to remove.

   Below is a sample data produced by the SQL query. The result should help you determine which entities have duplicate external-ids, and their external-id values and namespaces.

   And the external identifier can be removed using: `DELETE {entity-type}/{entity-id}/{identifier-value}`.

```
     c_entity_type     | c_entity_id | c_identifier_value | c_identifier_name
-----------------------+-------------+--------------------+------------------
↪-
 com.vidispine.db.Item | VX-11       | my_test_id         | namespace_1
 com.vidispine.db.Item | VX-2704     | my_test_id         | namespace_2
(2 rows)
```

   Automatic database migration can be performed after all duplicates have been removed.

### 18.2.5 Upgrading from 5.1 to 5.2

- Solr: No changes to the documents. Re-indexing is not required.

### 18.2.6 Upgrading from 5.0 to 5.1

- APIinit is needed due to a new job step for raw and essence imports.
- Solr: No changes to the documents. Re-indexing is not required.

### 18.2.7 Upgrading from 4.17 to 5.0

**Warning:** Because of a change in our internal system for the handling of database migrations, all users **MUST** upgrade to version 4.17 of Vidispine server and complete the database migration process before attempting to upgrade to Vidispine Server 5.0.

- Reindex needed due to Solr/Elasticsearch upgrade and changes to how dataset field values are indexed.
- APIinit is needed due to new roles being added.
- Solr: Supported version has been changed to 8.1.0.
- Elasticsearch: Supported version has been changed to 6.8.1.
- ActiveMQ: Supported version has been changed to 5.15.9.

For more information on upgrading to 5.0, see *Upgrading to Vidispine 5.0*.

### 18.2.8 Upgrading from 4.16 to 4.17

- APIinit is needed due to new metadata fields added for TTML parsing.
- Solr: No changes to the documents. Re-indexing is not required.

### 18.2.9 Upgrading from 4.15 to 4.16

- Solr: No changes to the documents. Re-indexing is not required.

### 18.2.10 Upgrading from 4.14 to 4.15

- Solr: No changes to the documents. Re-indexing is not required.
- The type of job that is created for the *placeholder raw import request* has changed from THUMBNAIL/TRANSCODE to PLACEHOLDER_IMPORT.

### 18.2.11 Upgrading from 4.13 to 4.14

- Solr: No changes to the documents. Re-indexing is not required.
- Elasticsearch: Supported versions have been changed to 5.6.x, 6.0.x, 6.1.x, or 6.2.x.
- The item re-index process will no longer also rebuild the thumbnail index that is maintained internally by VS. The thumbnail index can now instead be rebuilt separately using a *re-index thumbnail request*. To restore the old behaviour, set `disableThumbnailReindexing` to `false`.
- A number of new job steps have been added to support checksum validation on transfer.
  - COPY_FILE/MOVE_FILE, step 90 - Waiting for source file hash.
  - COPY_FILE/MOVE_FILE, step 92 - Retrieving source file hash.
  - COPY_FILE/MOVE_FILE, step 140 - Waiting for destination file hash.
  - COPY_FILE/MOVE_FILE, step 150 - Verifying file hashes.
- Previous versions had a bug where collections containing libraries did not always have recursive ACL's properly applied items in those libraries. This is fixed for newly created ACL's, but any old ones with this problem will need to be *re-indexed* with:

```
PUT /reindex/acl
```

- A role have been added for reading Vidispine Agents: `_vxa_read`. To make sure admin users receives the role properly, please run APIinit:

```
POST /APIinit
```

### 18.2.12 Upgrading from 4.12 to 4.13

- Solr: No changes to the documents. Re-indexing is not required.

### 18.2.13 Upgrading from 4.11 to 4.12

- Solr: No changes to the documents. Re-indexing is not required.
- All media shape deductions from jobs are now performed using asynchronous job steps. A number of new job steps have been added to support this.
  - TRANSCODE, step 400 - Finalizing media check.
  - CONFORM, step 400 - Finalizing media check.
  - AUTO_IMPORT, step 550 - Finalizing media check.
  - AUTO_IMPORT, step 1100 - Finalizing media check.
  - ESSENCE_VERSION, step 800 - Finalizing media check.
  - TIMELINE, step 400 - Creating the entities.
- To support removal of old essence files, a new job step has been added:
  - SHAPE_IMPORT, step 1000 - Remove old essence files.
- Make sure to run *APIinit* when upgrading to create the above mentioned job steps. Any existing custom job steps using these step numbers will be overwritten, so make sure to adjust the step number of any custom job step definitions using these numbers and recreate the custom steps.

### 18.2.14 Upgrading from 4.10 to 4.11

- Solr: No changes to the documents. Re-indexing is not required.

### 18.2.15 Upgrading from 4.9 to 4.10

- Solr: No changes to the documents. Re-indexing is not required.

### 18.2.16 Upgrading from 4.8 to 4.9

- Solr: No changes to the documents. Re-indexing is not required.
- Support for Ubuntu 12.04 has been discontinued.

### 18.2.17 Upgrading from 4.7 to 4.8

- Solr: No changes to the documents. Re-indexing is not required.

### 18.2.18 Upgrading from 4.6 to 4.7

- Solr: The Solr schema must be updated to use the new *long integer datatype*.
- Solr: No changes to the documents. Re-indexing is not required.
- Version 3.2 of the MatrixStore SDK is now installed by default. The *vidispine-server-matrixstore3.1* package must be installed to connect to MatrixStore 3.1.

### 18.2.19 Upgrading from 4.5 to 4.6

- Solr: No changes to the documents. Re-indexing is not required.
- Support for GlassFish has been discontinued. Use *Installation* instead.
- Support for Java 7 has been discontinued. Use Java 8 instead.
- Support for Microsoft Windows Server has been discontinued.

### 18.2.20 Upgrading from 4.4 to 4.5

- Solr: No changes to the documents. Re-indexing is not required.
- Support for JBoss has been discontinued. Use *Installation* instead.

### 18.2.21 Upgrading from 4.3 to 4.4

- Solr: No changes to the documents. Re-indexing is not required.
- The property `indexCollectionItemOrder` is now set to `false` by default. Set it to `true` before upgrading if you rely on the old behaviour.
- The transcode preset script is now also evaluated for conform jobs. The difference compared to transcodes is that the shape is empty. If you use the same preset for both transcodes and conforms, then make sure that the script verifies the existence of the components of the shape before using them to avoid null pointer exceptions from the script.

### 18.2.22 Upgrading from 4.2 to 4.3

- Solr: No changes to the documents. Re-indexing is not required.

### 18.2.23 Upgrading from 4.1 to 4.2

- Solr: The indexing of items, collections and files has changed. *Re-indexing is required.*
- `Essence version` resource representation has changed from URIListDocument to *EssenceVersionListDocument*, and now returns additional information about the version.

Platform:

- Java 7u67 is now supported with GlassFish. The installer must be used to install this update onto GlassFish, as configuration and libraries in GlassFish must be patched to work with Java 7u67.
- The installer will now install Solr 4.10.0 instead of Solr 4.5.1, but Solr 4.5.1 is still supported. Large systems may benefit of upgrading to Solr 4.10.0 as it brings additional search performance improvements.

### 18.2.24 Upgrading from 4.0 to 4.1

- The ObjectMatrix MatrixStore client SDK has been updated from 2.7.2.6 to 3.1.3.3. This version is not compatible with older MatrixStore server versions, so verify that you are running MatrixStore server 3.1.x before upgrading to Vidispine 4.1.5 or later.
- Support for Ubuntu 10.04 LTS has been discontinued, use Ubuntu 12.04 LTS instead.
- On Linux the number of allowed open files per process (File Descriptor Limit) must be raised from the default (1024) to at least 50000 to avoid Java IO errors `java.io.IOException: Too many open files`. See How to increase File Descriptor Limit on Linux (http://www.vidispine.com/partner/knowledge-forum-support/kb/how-tos/how-to-increase-file-descriptor-limit-on-linux).

# 18.3 Version 21.3

## 18.3.1 21.3

*2021-08-23*

### New versioning scheme for VidiCore and VidiCoder

In order to align versions within the Vidispine product portfolio, the versioning scheme for VidiCore and VidiCoder has been changed to `YEAR.QUARTER[.PATCH]`, e.g., `21.3`, `21.3.1`, `21.4`, and `22.1`. Users can still upgrade from any `5.x` to `21.3` directly.

### VidiNet Cognitive Service support in VidiCore

A number of new features and improvements has been added to support VidiNet Cognitive Service (VCS) on VidiNet:

- *Train a collection into a model using VidiNet Cognitive Service*.
- *Be able to output all items in a collection to a file using a LIST_ITEMS job*.
- *Add a callback location resources support in ANALYZE jobs*.

For more information about how to setup and use VCS, please refer to the documentation on VidiNet.

### Enhanced AWS IAM role support

User is now able to specify a `roleARN` parameter when setting up a AWS resource in VidiCore. It makes cross account access using IAM role possible. This is supported in *S3 storages method*, *S3 Event SQS Notifications*, *SQS*/*SNS* notification action

### Proxying HTTP connection via a VSA

It's now possible to *use VSA as a HTTP proxy*, so that the `http` javascript object in VidiCore is able to access on-prem resources:

```
http.uri("http://localservice:7777/integration")
      .proxy("vxa://742c17e4-8f6f-489a-8caf-aae4c39d272f/")
      .proxy("vxa://e2c4c766-4a3e-4662-975c-7f4251385d8c/")
      .get()
```

### Improvements

- Add external id when creating placeholder item (#4789).
- Support for AWS Restore Notifications (S3:ObjectsRestore:Completed) for SQS and SNS (#4723).
- Doing quota calculation in small chunks (#4931).
- Return 503 instead of 401 when the database is down (#4910).
- Include database in the health check endpoint: `/APInoauth/is-online?checkdb=true` (#4798).
- Implementation of Wildcard/Regex within the path logs in Audit Trails (#4625).
- Add a new `collection/{id}/collection` endpoint to fetch child collections (#4017).

### Transcoder improvements

- Allow pass-through of URIs in SequenceDocument to NLEJobDocuments (#4845).

### Bug fixes

- Possible integer overflow during timecode comparison (#4924).
- VSA connections may lead to race conditions (#4909).

## 18.4 Version 5.7

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.4.1 5.7.1

*2021-07-13*

#### Improvements

- Add support for rendering 59.94 framerate video timelines (#4920).
- Improve IMF 2020 XSD support (#4913).
- Improve deletion speed of items whose metadata is referenced in many places (#4912).
- Improve metadata update speed if it's referenced by a large number of entities (#4897).
- Indexing optimization of high-load metadata patterns (#4894).
- Support parsing FinalCut xml with sequence reference (#4887).
- Be able to parallelize bulky metadata migration (#4846).

#### Bug fixes

- Database migration error on MySQL and SQL Server (#4917).
- Incorrect SCC ROLL_UP parsing (#4903).
- Job finished notifications not been processed for a long time (#4900).

### 18.4.2 5.7

*2021-05-27*

#### Baton QC Service Support on VidiNet

It's now possible to use the Baton QC service if you are running VidiCore on VidiNet. For more information, please refer to VidiNet (https://vidinet.net/). (#4840)

#### New features

- *MacCaption support* (#4762).

#### Improvements

- Indexing and search performance improvements (#4881).
- Improve metadata retrieval speed with inherited metadata (#4849).
- Make it possible to choose directory traversing algorithm (#4824).
- Optimize job startup speed (#4808).

- Be able to use the storage external id when registering a file (#4786).
- Support IMF schema ST 2067-2:2020 (#4741).
- Be able to set *com.vidispine.service.quorum* via configuration properties (#4728).
- Only start decoder when it is actually needed in NLEJob (#4027).

#### Bug fixes

- VSA cannot connect to Spectra BlackPearl share (#4851).
- Search Cursor changes each time at the end of the search result (#4823).
- Not able to search for metadata field group with space in name (#4758).
- Fix missing *Content-Length* header from the APInoauth thumbnail and poster endpoints (#4743).
- Facet search not returning results in timed metadata (#4734).
- File retains inherited deletion-lock even after detached from item/shape (#4708).
- Fix incorrect Waveform URI not using the *apiNoauthURI* setting (#4624).

#### Transcoder fixes

- Memory leak regression in deinterlacefilter (#4864).
- Item sequence render conform job gives unexpected result when average framerate is off (#4784).
- Scenechange thumbnailPlugin results in thumbnails with SAME scene (#4337).

#### Upgrading from 5.6

- Solr: No changes to the documents. Re-indexing is not required.

## 18.5 Version 5.6

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.5.1 5.6.2

*2021-07-06*

#### Improvements

- Improve IMF 2020 XSD support (#4913).
- Improve deletion speed of items whose metadata is referenced in many places (#4912).
- Improve metadata update speed if it's referenced by a large number of entities (#4897).
- Support parsing FinalCut xml with sequence reference (#4887).
- Be able to parallelize bulky metadata migration (#4846).

#### Bug fixes

- Job finished notifications not been processed for a long time (#4900).

## 18.5.2 5.6.1

*2021-05-21*

### Improvements

- Indexing and search performance improvements (#4881).
- Improve metadata retrieval speed with inherited metadata (#4849).
- Make it possible to choose directory traversing algorithm (#4824).
- Optimize job startup speed (#4808).
- Be able to use the storage external id when registering a file (#4786).
- Support IMF schema ST 2067-2:2020 (#4741).
- Be able to set *com.vidispine.service.quorum* via configuration properties (#4728).

### Bug fixes

- VSA cannot connect to Spectra BlackPearl share (#4851).
- Search Cursor changes each time at the end of the search result (#4823).
- Not able to search for metadata field group with space in name (#4758).
- Fix missing *Content-Length* header from the APInoauth thumbnail and poster endpoints (#4743).
- Facet search not returning results in timed metadata (#4734).
- File retains inherited deletion-lock even after detached from item/shape (#4708).
- Fix incorrect Waveform URI not using the *apiNoauthURI* setting (#4624).

### Transcoder fixes

- Memory leak regression in deinterlacefilter (#4864).
- Item sequence render conform job gives unexpected result when average framerate is off (#4784).

## 18.5.3 5.6

*2021-04-27*

### Thumbnail sprite sheet

A *thumbnail sprite sheet* is a big image of all thumbnails laid out on a canvas, together with a description on where every thumbnail is stored. (#4794)

The sprite sheet can be used by UI to quickly navigate through all thumbnails without having to load them one by one.

Example:

```
GET /item/VX-14836/thumbnail/spritesheet
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ThumbnailSpriteSheetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
 <etag>a0231b88172bb02e77940fd118ead3f8</etag>
 <url>http://localhost:8080/API/thumbnail/VX-1/VX-14836;version=0/0?type=tsi&amp;
 ↪hash=5738287a0bdbd31c228263e74d250567</url>
 <path>VX-1/VX-14836;version=0/0?type=tsi&amp;hash=5738287a0bdbd31c228263e74d250567</
 ↪path>
```

```
 <thumbnail>
   <width>320</width>
   <height>234</height>
   <x>0</x>
   <y>0</y>
   <timecode>
     <samples>50</samples>
     <timeBase>
       <numerator>1</numerator>
       <denominator>50</denominator>
     </timeBase>
   </timecode>
   <endTimecode>
     <samples>250</samples>
     <timeBase>
       <numerator>1</numerator>
       <denominator>50</denominator>
     </timeBase>
   </endTimecode>
 </thumbnail>

 <!-- more thumbnails-->
 ...

</ThumbnailSpriteSheetDocument>
```

### Self-refreshing library support for Elasticsearch

*Self refreshing libraries* (`autoRefresh=true`) is now supported for the Elasticsearch *search backend*. This is implemented using the Elasticsearch percolate query (https://www.elastic.co/guide/en/elasticsearch/reference/6.8/query-dsl-percolate-query.html). (#3596)

A separate Elasticsearch index called `vidispine_percolator` is used to hold the percolate queries. It's created as part of the `vidispine elasticsearch init` command. Systems upgrading from older VidiCore versions can use `vidispine elasticsearch init --percolator-index` to only create the percolator index without touching the main index.

### Indexing speed improvement for Solr and Elasticsearch

Various fixes has been made to adjust the requests that VidiCore sends to Solr and Elasticsearch, in order to improve the indexing speed. (#4777, #4850)

For Elasticsearch, it's now possible to use multiple threads to send documents to Elasticsearch (see *`elasticsearchWorkerCount`*). It should improve the indexing speed while using Elasticsearch cluster. (#4701)

There is also a property *`elasticsearchBulkBuffer`*, which can be used to adjust the internal buffer size of the Elasticsearch worker thread. It will affect indexing speed as well.

### Different handling of timecodes from scc subtitle files

Before 5.6, the timecodes of scc subtitle metadata are taken directly from the subtitle file without modification. (#4314)

In 5.6, the default behavior has been changed: the result of `item.startTimecode -scc timecode` will be applied to the item metadata instead. If you want to retain the old behavior, *`useAbsoluteSccTimeCode`* has to be set to `true`.

### Possibility to override shape tag in sequence render request

When rendering a sequence, the codecs and settings in the requested shape tag definition can be *overridden by the SequenceDocument*. (#4506, #4507).

### New features

- Support for s3 pre-signed URL requests for upload with KMS-SSE buckets (#4628).
- Add support for file lost notification (#4711).
- Decode support for CR3 (Canon Raw) format (#4834).
- Decode support for HEIC images (#4698).
- Set ChannelLayout for the transcoded output (#4559).
- Extend mix element to hold time-based mix information (#4506).

### Improvements

- Make message acknowledgement faster when ActiveMQ is configured to use JDBC datasource (#4778).
- Be able to hash files on a VidiCore S3 storage using VSA (#4694).
- Implement support for cost estimates for hightscreen/smartcrop jobs (#4654).
- Support for multiple publickeys & certificates in Oauth2 config (#4595).
- Update VSCTL to Python3 (#4586).
- Always start/restart transcoder service after install/update (#4724).

### Bug fixes

- Possible broken notification documents after a *vacuumlo* on PostgreSQL (#4831).
- Invalid PUT request when closing files (#4803).
- Double indexing on item placeholder creation (#4776).
- Not possible to add vsa network share where host contains port. (#4759).
- Slow reindex of collections compared to items (#4680).
- VSA add-network-share doesn't work with Scality S3 (#4363).
- VSA allows access/import of files outside local fs share (#4670).

### Transcoder fixes

- Shape standard properties are not consistent after transcoding (#4795).
- Audio gain not working properly (#4767).
- Audio fade out using AudioCrossFade is not working (#4749).
- Add support for transition at start of timeline (#4745).
- FadeInOutDissolve not working (#4727).
- Lip-sync issue when transcoding a file with lowres shape-tag (#4700).
- Error when transcoding PDF file with different page size and orientation (#3943).

#### Upgrading from 5.5

- Solr: No changes to the documents. Re-indexing is not required.

## 18.6 Version 5.5

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.6.1 5.5.3

*2021-07-05*

#### Improvements

- Improve IMF 2020 XSD support (#4913).
- Improve deletion speed of items whose metadata is referenced in many places (#4912).
- Improve metadata update speed if it's referenced by a large number of entities (#4897).
- Be able to parallelize bulky metadata migration (#4846).

#### Bug fixes

- Job finished notifications not been processed for a long time (#4900).

### 18.6.2 5.5.2

*2021-05-21*

#### Improvements

- Improve metadata retrieval speed with inherited metadata (#4849).
- Make it possible to choose directory traversing algorithm (#4824).
- Optimize job startup speed (#4808).
- Be able to use the storage external id when registering a file (#4786).
- Support IMF schema ST 2067-2:2020 (#4741).
- Be able to set *com.vidispine.service.quorum* via configuration properties (#4728).

#### Transcoder improvements

- Always start/restart transcoder service after install/update (#4724).

#### Bug fixes

- VSA cannot connect to Spectra BlackPearl share (#4851).
- Search Cursor changes each time at the end of the search result (#4823).
- Itemsearch bug when searching for metadata field group name with space (#4758).
- Fix missing *Content-Length* header from the APInoauth thumbnail and poster endpoints (#4743).
- Facet search not returning results in timed metadata (#4734).
- File retains inherited deletion-lock even after detached from item/shape (#4708).

#### Transcoder fixes

- Memory leak regression in deinterlacefilter (#4864).
- Item sequence render conform job gives unexpected result when average framerate is off (#4784).

### 18.6.3 5.5.1

*2021-03-26*

#### Improvements

- Allow creating storage methods with same URI but different method types (#4131).

#### Bug fixes

- Cannot add metadata when creating a new shape (#4738).
- Transcode job with "scenechange" thumbnail plugin not working on Vidinet (#4736).
- Duplicate storage-rule file action jobs are stated in some case (#4710).
- parameter importTag not working with raw import (#4695).
- Restarting VSA removes transcoder directAccess element (#4677).
- "db check" command should fail if the database is empty (#4668).
- Incorrect facet results for Elasticsearch, if the SearchDocument contains a "filter" (#4666).
- The AUTO-VSA url generate by VidiCore is incorrect (#4663).

#### Transcoder fixes

- Audio/video(frozen video) issues after a conform job for .mxf files (#4735).
- Audio/video(unsynced audio) issues after a conform job for .mov (#4733).
- Cannot extract closed captions in MXF file (#4721).
- Transcode job no failing after a file reading error (#4707).
- Conform Job with Audio Mix Option fails with empty mi (#4692).
- Incorrect samples count and duration for aac (#4681).
- Transcoder license endpoint race condition (#4618).

#### Agent fixes

- Incorrect analyze job result if using VSA (#4706).

### 18.6.4 5.5

*2021-01-29*

#### Collection-to-collection joins

Collection *hierarchy joins* can be used to find collections based on their relationship with other collections (#4614).

For example, to find collections that have a child collection with `title=vidispine`.

```
PUT /collection
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <collection>
    <field>
      <name>title</name>
      <value>vidispine</value>
    </field>
  </collection>
</ItemSearchDocument>
```

### Collection improvements

An item can now be stored *multiple times in the same collection* (#4615). Metadata on the relation can be used do define for example that only a specific segment from the item is considered part of the collection.

### New features

- Return timespans which *match search* using `interval=result` (#4648).
- Searching using cursors is now supported for non-generic timespans. Note some *difference in functionality for Elasticsearch* (#4308).

### Bug fixes

- Shape deduction of EPS file gives binary component (#4704).
- Shape subquery does not work when searching for collection (#4642).
- Using role query param in group search returns 500 (#4616).
- Exception when deleting document metadata by UUID (#4613).
- Global metadata update failing due to slow metadata query (#4604).
- Slow deletion on systems with a large amount of deletion locks (#4588).
- Request failures when performing raw import using concurrent multipart uploads (#4554).
- Metadata group gets created multiple times with same name in the metadata schema (#4318).
- Incorrect auto commit setting in Solr package (#4313).
- Bulky metadata not removed on analyze job failure (#4153).
- Incorrect service “isRunning” flag in a cluster setup (#3890).

### Transcoder fixes

- Incorrect image output for PSD to JPEG/PNG transcode (#4672).
- Transcoder segfault for ProRes when pixel format is empty or unsupported (#4589).

### Agent fixes

- Read timed out error in conform jobs via VSA (#4582).

### Upgrading from 5.4

- APIinit is needed due to a new job step for analyze jobs.
- Solr: No changes to the documents. Re-indexing is not required.
- This release contains database migrations for both metadata and collections that may take time on larger systems. Please plan upgrades accordingly.

### Breaking changes

- Searching with cursors: Previously, when reaching the end of a result set, the *next* cursor returned was null. With 5.5 a valid cursor is returned instead. This enables tailing searches (https://lucene.apache.org/solr/guide/8_0/pagination-of-results.html#tailing-a-cursor). To check for end of search, either check hits being less than the requested number, or if the returned cursor is the same as the previous cursor.
- When enabling `indexCollectionItemOrder`, the order is in ascending order by the first entry of the item in the respective collection. (It is still recommended that this property is set to `false` for performance reasons.)

## 18.7 Version 5.4

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.7.1 5.4.5

*2021-07-05*

#### Improvements

- Improve IMF 2020 XSD support (#4913).
- Improve deletion speed of items whose metadata is referenced in many places (#4912).
- Improve metadata update speed if it's referenced by a large number of entities (#4897).
- Be able to parallelize bulky metadata migration (#4846).

#### Bug fixes

- Job finished notifications not been processed for a long time (#4900).

### 18.7.2 5.4.4

*2021-05-21*

#### Improvements

- Optimize job startup speed (#4808).
- Be able to use the storage external id when registering a file (#4786).
- Support IMF schema ST 2067-2:2020 (#4741).
- Be able to set *com.vidispine.service.quorum* via configuration properties (#4728).

#### Transcoder improvements

- Always start/restart transcoder service after install/update (#4724).

#### Bug fixes

- VSA cannot connect to Spectra BlackPearl share (#4851).
- Itemsearch bug when searching for metadata field group name with space (#4758).
- Fix missing *Content-Length* header from the APInoauth thumbnail and poster endpoints (#4743).
- Facet search not returning results in timed metadata (#4734).
- File retains inherited deletion-lock even after detached from item/shape (#4708).

#### Transcoder fixes

- Memory leak regression in deinterlacefilter (#4864).
- Item sequence render conform job gives unexpected result when average framerate is off (#4784).

### 18.7.3 5.4.3

*2021-03-26*

#### Improvements

- Allow creating storage methods with same URI but different method types (#4131).

#### Bug fixes

- Cannot add metadata when creating a new shape (#4738).
- Transcode job with "scenechange" thumbnail plugin not working on Vidinet (#4736).
- Duplicate storage-rule file action jobs are stated in some case (#4710).
- parameter importTag not working with raw import (#4695).
- Restarting VSA removes transcoder directAccess element (#4677).
- "db check" command should fail if the database is empty (#4668).
- Incorrect facet results for Elasticsearch, if the SearchDocument contains a "filter" (#4666).
- The AUTO-VSA url generate by VidiCore is incorrect (#4663).

#### Transcoder fixes

- Audio/video(frozen video) issues after a conform job for .mxf files (#4735).
- Audio/video(unsynced audio) issues after a conform job for .mov (#4733).
- Transcode job no failing after a file reading error (#4707).
- Incorrect samples count and duration for aac (#4681).
- Transcoder license endpoint race condition (#4618).

#### Agent fixes

- Incorrect analyze job result if using VSA (#4706).

### 18.7.4 5.4.2

*2021-01-25*

### Bug fixes

- Item still inherits deletion-lock from collection after removing (#4673).
- Effective lock change notification is not triggered when item is added into a collection (#4436).
- Prevent multiple services from starting if cluster is in a "split-brain" status (#4653).
- Removed metadata field shows up again after trimming (#4646).

### Transcoder fixes

- Transcoder does not allocate large enough moov header (#4674).
- Slow shape deduction of large MXFs due to large block size (#4675).
- Transcoder not burning in text overlay (#4682).
- New mutex is created instead of locking existing mutex (#4679).

### Agent fixes

- Failed to rename files in a VSA share whose URI contains query parameters (#4609).

## 18.7.5 5.4.1

*2020-12-11*

### Bug fixes

- VCS analysis requests rejected with HTTP 400 (#4661).
- Collection not re-indexed when a child item is deleted (#4629).
- Files generated by a retried transcode step is not linked to container component (#4603).
- HTTP 500 when adding S3 storage method with region parameter (#4163).
- Warning in logs: SearchBean.searchItemsByIds() modifies input arguments (#4544).
- Item metadata cannot be returned as expected if queries contain group (#4637).
- Restoring large file from Glacier vault results in IOException (#4638).
- Audit trails body field is truncated after few hundred chars (between 700-1100) (#4636).
- Image import fails due to ArrayIndexOutOfBoundsException (#4605).
- JobCruncherWorker flooding logs if ActiveMQ is not ready (#4020).
- Incorrect link to support portal on Welcome page (#4610).
- Swedish characters not displayed correctly when transferred via FTP (#4477).
- JavaScript objects have too strict argument type checks with GraalJS (#4214).
- Missing field default values while accessing collection metadata (#4461).
- Filter on groups does not work for the /search endpoint (#4562).
- Deleted metadata field still appears in metadata (#4411).
- Item not re-indexed after being associated with a file (#4449).
- NPE when restoring file from Glacier to S3 (#4187).

#### Transcoder fixes

- Burned in .stl subtitles produce artifacts (#4656).
- Output frame calculation does not respect FPS change (#4593).
- Sample format AV_SAMPLE_FMT_S32 not supported error when encoding AAC (#4587).
- H264_FIELDBASED InterlaceMode setting not applied for Nablet (#4632).

### 18.7.6 5.4

*2020-11-02*

#### VidiNet

#### Nablet Shrynk and Heightscreen

Support for the upcoming Nablet Shrynk and Nablet Heightscreen services in VidiNet has been added. This will allow you to *create highlight reels* and *perform smart cropping* of your videos (#4599, #4600).

#### MPEG-DASH

Support for transcoding into *MPEG-DASH* using VidiNet has been added (#4431). This allows you to create MPEG-DASH representations with varying bit rate and resolution. The representations, along with a manifest, can be exported using an *export job*.

#### Storage scans

For large storages it can sometimes be useful to only rescan a specific portion of it. This can now be done by triggering a *storage rescan with a prefix specified* (#4091).

#### Job improvements

When using custom jobs and/or custom JavaScript job steps to implement a workflow, there can be jobs that need to start and/or wait for other jobs to finish before proceeding. For example, a custom export job may need to wait for a transcode job to finish.

To avoid delays due to excessive waiting, custom jobs can now *wait on other jobs* using the new `job.waitForJobs()` function (#4360).

For example:

```
job.waitForJobs('Waiting for jobs to finish...', jobIds);
```

#### Dedicated job pools

The job pool configuration has also been updated to support having job pools that always execute jobs of a certain priority, by enabling *dedicated job pools* (#4356).

#### New features

- New endpoint to see the *access controls a user has* (#4328).
- Implement cursor support on search endpoint for generic timespan only (#4306).

### Improvements

- Restrict fields when *viewing full change sets history* (#3884).
- Make *transaction timeout configurable* (#4581).
- Make *Elastic APM max spans configurable* (#4580).
- Resolve metadata references once for faster metadata updates (#4583).
- Performance improvement on validating metadata with a large dataset (#4534).
- Link files generated by a retried PLACEHOLDER_IMPORT transcode step to shapes sooner (#4513).
- Only allow users with _group_write to manage roles they already have (#4322).
- Include empty groups in the *applied access controls list for groups* (#4285).

### Transcoder improvements

- Support setting opacity for overlays (#3738).
- Provide Transcoder with systemd unit file (#4332).

### Agent improvements

- Make *transfer jobs* and *hash jobs* in VSA multi-threaded (#4476).

### Bug fixes

- Scenarist parser lacks support for roll up 2 action code (#4560).
- Deleting a job does not remove the jobs external ids (#4556).
- Closed captions (CEA-608) not extracted from MXF data tracks (#4503).
- Time out error in “dbstats” selftest (#4186).

### Transcoder fixes

- Transcoder hangs on CentOS 8 on certain image jobs (#4601).
- Cannot extract embedded 608 / 708 closed captions for MXF files containing DNX HD145 (#4596).
- Color shift issues after transcode (#4528).

### Platform

- This release adds support for Ubuntu 20.04.
- The server and agent tools now depend on Python 3 (#4501).

### Upgrading from 5.3

- Solr: No changes to the documents. Re-indexing is not required.

## 18.8 Version 5.3

The release notes will tell you what’s new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.8.1 5.3.6

*2021-03-26*

#### Bug fixes

- "db check" command should fail if the database is empty (#4668).
- Incorrect facet results for Elasticsearch, if the SearchDocument contains a "filter" (#4666).

#### Transcoder fixes

- Audio/video(frozen video) issues after a conform job for .mxf files (#4735).
- Audio/video(unsynced audio) issues after a conform job for .mov (#4733).
- Transcode job no failing after a file reading error (#4707).
- Incorrect samples count and duration for aac (#4681).
- Transcoder license endpoint race condition (#4618).

### 18.8.2 5.3.5

*2021-01-22*

#### Bug fixes

- Effective lock change notification is not triggered when item is added into a collection (#4436).
- Prevent multiple services from starting if cluster is in a "split-brain" status (#4653).
- Removed metadata field shows up again after trimming (#4646).

#### Transcoder fixes

- Slow shape deduction of large MXFs due to large block size (#4675).
- Transcoder does not allocate large enough moov header (#4674).
- New mutex is created instead of locking existing mutex (#4679).

#### Agent fixes

- Failed to rename files in a VSA share whose URI contains query parameters (#4609).

### 18.8.3 5.3.4

*2020-12-11*

#### Bug fixes

- Item metadata cannot be returned as expected if queries contain group (#4637).
- Restoring large file from Glacier vault results in IOException (#4638).
- Audit trails body field is truncated after few hundred chars (between 700-1100) (#4636).
- Image import fails due to ArrayIndexOutOfBoundsException (#4605).

### Transcoder fixes

- Output frame calculation does not respect FPS change (#4593).
- Sample format AV_SAMPLE_FMT_S32 not supported error when encoding AAC (#4587).

## 18.8.4 5.3.3

*2020-11-05*

### Bug fixes

- Time out error in "dbstats" selftest (#4186).
- JobCruncherWorker flooding logs if ActiveMQ is not ready (#4020).
- Item order on collection search not same as order in collection (#4611).
- Incorrect indexing after metadata field creation/update (#4612).
- Incorrect link to support portal on Welcome page (#4610).
- Swedish characters not displayed correctly when transferred via FTP (#4477).
- JavaScript objects have too strict argument type checks with GraalJS (#4214).
- Missing field default values while accessing collection metadata (#4461).
- Filter on groups does not work for the /search endpoint (#4562).
- Deleted metadata field still appears in metadata (#4411).
- Item not re-indexed after being associated with a file (#4449).
- NPE when restoring file from Glacier to S3 (#4187).

### Transcoder fixes

- H264_FIELDBASED InterlaceMode setting not applied for Nablet (#4632).

## 18.8.5 5.3.2

*2020-10-21*

### Improvements

- Support TLS protocol and certificate setting for LDAP connections (#4551).
- Include fields with default values in the search index (#4464).
- Storage rules should allow for new storage actions while processing the current batch (#4533).
- Expose *storage class* for files on S3 Glacier Deep Archive (#4201).

### Bug fixes

- FTP control connection not kept alive while file transfer is in progress (#4388).
- Preset with both mix and stream elements do not render properly (#4606).
- Slow collection update of large collections due to synchronous indexing (#4520).
- Unnecessary reindexing of files on VSA if scan fails and resumes (#4567).
- Possible ACL reindex failure after transaction failure (#4563).

- Metadata dataset validation fails for values containing slashes (#4522).
- Glacier archival fails due to timeout exception (#4594).
- Glacier vault storage doesn't handle the retrievalTier from storage metadata (#4552).
- NullPointerException in GlacierBean (#4542).
- Collection not reindexed after being renamed (#4525).
- Collection metadata not validated for POST API/collection (#4493).
- Incorrect transient metadata for item if file copy job fails (#4174).
- Incorrect deletion lock metadata for copied file (#3825).
- User can assign himself to a group without _group_write permission (#4546).
- Imports failing due to Infinispan lock errors for XMP fields (#4487).
- Bulk collection delete fails if collection has ancestor relationship (#4405).
- Direct S3 to S3 transfers not used when useSegmentFiles is true (#4383).
- Metadata modified notification triggered on empty changes (#4465).
- Renamed file not re-detected until after 5 hours (#4579).
- No write from transcoder in 1 hour causes mutable range write to fail (#4575).
- Error retrieving thumbnails from VSA (#4165).

#### Transcoder fixes

- Audio segments does not start at zero PTS (causing blanks in video) (#4607).
- Audio is broken for mov input files (#4536).
- Transcode of MP3 files fail with a license error (#4527).
- Distorted audio for Nablet AAC if sample format is planar (#4526).
- Transcoder does not allow multiple segments that are equal in timeline/CONFORM job (#4446).
- Missing timeCodeTimeBase element on container component (#4159).

### 18.8.6 5.3.1

*2020-08-30*

#### New features

- Support returning additional user/group information in ACL (#4505).
- Allow core metadata fields to be modified (#4482).

#### Improvements

- Send file delete notification if a lowres file is removed due to transcoding failure (#4499).
- Be able to configure the global spring async pool (#4495).

### Bug fixes

- Change the owner of an entity will generate a duplicate "indexLog" entry (#4450).

  Note: `db migrate` is needed to remove the already existing duplicate entries in `t_indexlog`. The migration is considered as *optional* since this issue doesn't cause any problem functionally. It only affects the performance of ACL reindex. You can choose when to run `db migrate`, and the duplicate entries won't be removed until then.

- Prevent possible JGroups "split-brain" issue during VidiCore startup (#4518).
- Mimetype missing on new shape versions (#4514).
- The output file of a "m4a" shape tag should have "m4a" as file extension (#4504).
- Reindex/index progress stuck on large collections (#4498).
- VidiCore fails to update VidiCoder License in some cases (#4494).
- Incorrect search hit on the */search* endpoint in some cases (#4488).
- Possible OOM when updating metadata referenced in many places (#4484).
- Glacier file with special character in name stops storage scan (#4472).
- Export between VSA shares takes long time and results in incorrect file size (#4460).
- Too many parameter error in SQL query from DB cleanup (#4438).
- Placeholder/Raw import jobs missing the configured CC extraction (#4421).
- Incorrect timecode if fetching metadata containing "inherited timespan" using the "interval" query parameter (#4368).
- Export with both tag and interval ignores the tag (#4156).
- Removed child collection is still available from parent collection search (#4404)
- Glacier Vault Archive restore job created by storage rule is aborted due to timeout (#4523)

### Transcoder fixes

- Creating video in 60 fps produces invalid length (#4481).
- Incorrect system metadata fields on item level for NTSC drop-frame videos (#4480).
- Fix invalid audio in MXF created by AVID (#4418).

## 18.8.7 5.3

*2020-07-15*

### New names

#### VidiCore

Vidispine Server, Vidispine API, VaaS are now all called VidiCore. The packages and names in documentation will change during the second half of 2020. Consequently, VSA and VMA now stands for VidiCore Server Agent and VidiCore Management Agent.

#### VidiCoder

The transcoder's new name is VidiCoder. Packages and documentation will change to reflect the new name during 2020.

### Breaking changes

- The `priority` attribute in the `MergedAccessDocument` has been changed to `rank` (#4430). Check *doc* for more information.
- If you're using external identifiers, database migration may require manual intervention. Check the "Upgrading from 5.2" section below for details.

### Features and improvements

### New file analyze endpoints

*Shape deduction* can now be performed on non-imported files or IMF packages in VidiCore (#4039).

```
POST /storage/{storage-id}/file/{file-id}/analyze
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-10525</jobId>
  <user>admin</user>
  <started>2020-07-14T09:04:51.768Z</started>
  <status>READY</status>
  <type>FILE_ANALYZE</type>
  <priority>MEDIUM</priority>
</JobDocument>
```

The resulting shape can be found in the job metadata and retrieved using the new *shape* endpoint on a file.

### Bulky metadata storage

By default, the values of *bulky metadata* are stored in the database. In a large system, this can occupy a large portion of the database. Now, it is possible to store bulky metadata on the file system (or cloud storage) (#4440). See *bulky metadata storage*.

### Configurable site prefix, identifier format, and logging level via the API

Traditionally, the VidiCore *site prefix* and *identifier format* can be configured using JVM properties. And the log levels are configured in the `server.yaml` file.

In 5.3, they can be configured using the API as well (#4417). See *doc*. This is especial useful to VidiCore users on VidiNet.

### Entity ownership transfer

The ownership of an entity can be transferred to a different user when the original owner is being deleted. Check the *usage* of the `transferAccess` parameter (#4327).

### VSA Thread configuration

It is now possible to configure the number of worker threads and selector runner threads in *agent properties* (#4226). This could improve VSA performance when under high load.

### VidiCoder (Transcoder) library update

The transcoder uses ffmpeg for some codecs. The ffmpeg libraries are bundled with the transcoder. In version 5.3, a major upgrade of the ffmpeg library is done. This means some ffmpeg-specific parameters may have be renamed. If you are using these parameters, let Vidispine now. We have also taken the opportunity to clean up some of the

edge-case handling of slightly incorrect media files in VidiCoder, as the new ffmpeg library is better at handling these. Hence, you should test that your type of media works with the new VidiCoder version. However, in the meantime, rest assured that you can still use VidiCoder 5.2 with VidiCore 5.3, they are compatible.

### Other improvements

- Add support for Bearer token validation using a public key (#4478).
- Implement support for import of MPEG-DASH package (#4379).
- Implement support for export of MPEG-DASH package (#4381).

### Bug fixes

- There can be duplicate external identifiers for the same entity type (#4350).
- Sidecar file cannot be imported for the second time (#4235).
- Glacier Vault storage can only get credentials from AwsCredentials.properties (#4219).

### Upgrading from 5.2

### External identifier migration

*External identifiers* in VidiCore should be unique among the same entity type. But if there are multiple "overlapping" external identifier namespaces defined, duplicate identifiers could exist.

This has been fixed in 5.3 (#4350). A stronger constraint has been added to the table to prevent duplicates. However, if there are already duplicate entries in the table, manual intervention is required.

Below are the steps:

1. Use this SQL statement to identify duplicated entries:

```
SELECT i2.c_entity_type, i2.c_entity_id, i2.c_identifier_value, i2.c_
↪identifier_name
FROM
(SELECT c_entity_type, c_identifier_value FROM t_externalidentry
    GROUP BY c_entity_type, c_identifier_value HAVING COUNT(*) > 1) i1
INNER JOIN
(SELECT * from t_externalidentry ) i2
ON i2.c_entity_type = i1.c_entity_type AND i2.c_identifier_value = i1.c_
↪identifier_value
ORDER BY c_identifier_value;
```

2. If the above query doesn't return anything, meaning no duplicates, you can skip the rest and processed with automatic database migration.
3. If there are duplicate entries, you need to decide which one(s) to remove.

   Below is a sample data produced by the SQL query. The result should help you determine which entities have duplicate external-ids, and their external-id values and namespaces.

   And the external identifier can be removed using: `DELETE {entity-type}/{entity-id}/{identifier-value}`.

```
     c_entity_type     | c_entity_id | c_identifier_value | c_identifier_name
-----------------------+-------------+--------------------+------------------
↪-
 com.vidispine.db.Item | VX-11       | my_test_id         | namespace_1
 com.vidispine.db.Item | VX-2704     | my_test_id         | namespace_2
(2 rows)
```

Automatic database migration can be performed after all duplicates have been removed.

## 18.9 Version 5.2

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.9.1 5.2.5

*2020-11-05*

#### Bug fixes

- Swedish characters not displayed correctly when transferred via FTP (#4477).
- JavaScript objects have too strict argument type checks with GraalJS (#4214).
- Missing field default values while accessing collection metadata (#4461).
- Filter on groups does not work for the /search endpoint (#4562).
- Deleted metadata field still appears in metadata (#4411).
- Item not re-indexed after being associated with a file (#4449).
- NPE when restoring file from Glacier to S3 (#4187).

#### Transcoder fixes

- H264_FIELDBASED InterlaceMode setting not applied for Nablet (#4632).

### 18.9.2 5.2.4

*2020-10-20*

#### Improvements

- Expose *storage class* for files on S3 Glacier Deep Archive (#4201).

#### Bug fixes

- Collection not reindexed after being renamed (#4525).
- Collection metadata not validated for POST API/collection (#4493).
- Incorrect transient metadata for item if file copy job fails (#4174).
- Incorrect deletion lock metadata for copied file (#3825).
- User can assign himself to a group without _group_write permission (#4546).
- Imports failing due to Infinispan lock errors for XMP fields (#4487).
- Bulk collection delete fails if collection has ancestor relationship (#4405).
- Direct S3 to S3 transfers not used when useSegmentFiles is true (#4383).
- Metadata modified notification triggered on empty changes (#4465).
- Renamed file not re-detected until after 5 hours (#4579).
- No write from transcoder in 1 hour causes mutable range write to fail (#4575).

- Error retrieving thumbnails from VSA (#4165).

### 18.9.3 5.2.3

*2020-08-31*

#### Bug fixes

- Change the owner of an entity will generate a duplicate "indexLog" entry (#4450).

  Note: `db migrate` is needed to remove the already existing duplicate entries in `t_indexlog`. The migration is considered as *optional* since this issue doesn't cause any problem functionally. It only affects the performance of ACL reindex. You can choose when to run `db migrate`, and the duplicate entries won't be removed until then.

- Prevent possible JGroups "split-brain" issue during VidiCore startup (#4518).
- Mimetype missing on new shape versions (#4514).
- The output file of a "m4a" shape tag should have "m4a" as file extension (#4504).
- VidiCore fails to update VidiCoder License in some cases (#4494).
- Incorrect search hit on the */search* endpoint in some cases (#4488).
- Possible OOM when updating metadata referenced in many places (#4484).
- Glacier file with special character in name stops storage scan (#4472).
- Too many parameter error in SQL query from DB cleanup (#4438).
- Incorrect timecode if fetching metadata containing "inherited timespan" using the "interval" query parameter (#4368).
- Export with both tag and interval ignores the tag (#4156).
- Removed child collection is still available from parent collection search (#4404)

#### Transcoder fixes

- Creating video in 60 fps produces invalid length (#4481).
- Incorrect system metadata fields on item level for NTSC drop-frame videos (#4480).
- Fix invalid audio in MXF created by AVID (#4418).
- Regressions in certain ProRes decodes since transcoder 4.13 (#3866).

### 18.9.4 5.2.2

*2020-06-18*

#### New features

- Add *maxConcurrency* setting to taskGroup (#4076).

#### Improvements

- Be able to choose which transcoder to use for an *essence import* job using the `resourceId` parameter.(#4458).
- Improved IMF support (#4427).
- Support case-insensitive sorting in search (#4406).
- Add 'vidinetJobId' and 'vidinetResourceId' to analyze job metadata (#4401).

#### Bug fixes

- Vidispine server fails to send/create essence version job to Vidinet (#4422).
- Deletion-lock document shows wrong information when long identifier is set (#4387).
- Reindexing fails if an "recursive=true" ACL is applied to an entity with a lot of sub-entities. (#4367).
- Empty thumbnail result when the "version" query parameter is "all" (#4310).
- Metadata not updated after deleting essence version (#4279).

#### Agent fixes

- Import from VSA where path contains whitespace fails (#4407).

### 18.9.5 5.2.1

*2020-05-08*

#### New features

- Global setting for default job type priority (#4351).
- Be able to configure concurrency limit by job type (#4076).

#### Improvements

- Filter out users that are being deleted from the result of "GET /user" (#4302).
- Be able to create projectVersion with file path containing spaces (#4361).
- Be able to set "duration" to 1 minute for pre-signed URLs (#4336).
- Closing raw file chunks should be asynchronous (#4238).
- Allow overlapping subtitle elements (#3847).

#### New transcoder features

- Extract SEI frame timestamp (#4325).

#### Bug fixes

- Thumbnails incorrectly removed after essence version deletions (#4344).
- Users can not write their own metadata via the metadata endpoint (#4338).
- NPE while closing a file if its storage contains a storage method with unsupported scheme (#4334).
- Cannot create collection with schema validation enabled without specifying bogus collection id (#4236).
- PUT request with empty body does not work from JavaScript (#4175).
- In some cases, thumbnails were not deleted after an item had been removed. (#4021).
- SCC import removes spaces around italic markers (#3737).

#### Transcoder fixes

- Overlays are not working properly for YUV422P16 pixel formats (#4286).

### Agent fixes

- Files with semicolon in path do not import using VSA (#4230).

## 18.9.6 5.2

*2020-04-20*

### CentOS 8

CentOS 8 is now supported (#4261). Packages can be found in the repository (http://repo.vidispine.com/).

### Notification improvements

It's now possible to *send notifications to an Amazon SNS* topic (#4149) using configurations like:

```
<NotificationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <action>
    <sns synchronous="false">
      <contentType>application/xml</contentType>
      <endpoint>sns.eu-west-1.amazonaws.com</endpoint>
      <topic>arn:aws:sns:eu-west-1:############:sns-topic-name</topic>
      <secret>aws</secret>
    </sns>
  </action>
  <trigger>
    ...
  </trigger>
</NotificationDocument>
```

Support for Amazon SQS FIFO (First-In-First-Out) queues has been added as well (#3453). See *here*.

### Support job facet counts

A `/job/search` *endpoint* has been added to support facet counts of jobs (#4229):

```
PUT /job/search
Content-Type: application/xml

<JobSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <facet count="true" name="JobTypes">
    <field>type</field>
  </facet>
</JobSearchDocument>
```

```
<JobListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <hits>2</hits>
    <job>
      <jobId>VX-1</jobId>
      <user>admin</user>
      <started>2020-02-19T13:53:21.110Z</started>
      <finished>2020-02-19T13:53:21.242Z</finished>
      <status>FINISHED</status>
      <type>RAW_IMPORT</type>
      <priority>MEDIUM</priority>
    </job>
    <job>
      <jobId>VX-2</jobId>
```

```
      <user>admin</user>
      <started>2020-02-19T13:53:21.110Z</started>
      <finished>2020-02-19T13:53:21.242Z</finished>
      <status>STARTED</status>
      <type>AUTO_IMPORT</type>
      <priority>MEDIUM</priority>
    </job>
    <facet name="JobTypes">
      <field>type</field>
      <count fieldValue="RAW_IMPORT">1</count>
      <count fieldValue="AUTO_IMPORT">1</count>
    </facet>
</JobListDocument>
```

### Improvements

- Paging support for user group search (#4162).
- Case insensitive user search (#4118).
- Filter users by enabled/disabled status (#4072).
- Add support for audio level adjustments on the NLEJob timeline (#4025).
- Support mixing of sequence renditions (#3848).
- Improve the performance of fetching metadata with "defaultValue=true" in a cluster environment. (#4319).
- Support SQL Query rewriting for MSSQL (#4324).
- Be able to configure the batch size in DeletionLockBufferCruncher. (#4099)

### Bug fixes

- Remove unused database indexes (#4280).
- Possible infinite retry of a job step (#4259).
- Slow indexing on system using external ids (#4101).
- Deletion-lock does not show full siteId in systems using long identifiers (#4195).
- EXPORT job is incorrectly marked as FINISHED if one of its steps fails due to being DISAPPEARED (#4200).
- Fix possible NegativeArraySizeException when uploading large files to S3 (#4256)
- Abort Vidispine job if transfer step throws a runtime exception (#4305)

### Transcoder fixes

- Incorrect results from CONFORM job on long GOP material (#3549).
- Deinterlacing provides wrong frame rate (#4258)
- Thumbnailbackground settings in TranscodePresetDocument does not take effect (#3909)
- Scaling is ignored in NLEJob (#4023)
- Subtitle rendering is broken for all video formats that are not 4:2:0 (#4024)
- Closed Captions(EIA-608,CEA-708) not found (#4246)

#### Upgrading from 5.1

- Solr: No changes to the documents. Re-indexing is not required.

## 18.10 Version 5.1

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.10.1 5.1.6

*2020-10-20*

#### Improvements

- Expose *storage class* for files on S3 Glacier Deep Archive (#4201).

#### Bug fixes

- User can assign himself to a group without _group_write permission (#4546).
- Imports failing due to Infinispan lock errors for XMP fields (#4487).
- Bulk collection delete fails if collection has ancestor relationship (#4405).
- Direct S3 to S3 transfers not used when useSegmentFiles is true (#4383).
- Metadata modified notification triggered on empty changes (#4465).
- Renamed file not re-detected until after 5 hours (#4579).
- No write from transcoder in 1 hour causes mutable range write to fail (#4575).
- Error retrieving thumbnails from VSA (#4165).

### 18.10.2 5.1.5

*2020-08-29*

#### Bug fixes

- Prevent possible JGroups "split-brain" issue during VidiCore startup (#4518).
- Mimetype missing on new shape versions (#4514).
- VidiCore fails to update VidiCoder License in some cases (#4494).
- Incorrect search hit on the */search* endpoint in some cases (#4488).
- Possible OOM when updating metadata referenced in many places (#4484).
- Too many parameter error in SQL query from DB cleanup (#4438).
- Incorrect timecode if fetching metadata containing "inherited timespan" using the "interval" query parameter (#4368).
- Export with both tag and interval ignores the tag (#4156).
- Removed child collection is still available from parent collection search (#4404)

#### Transcoder fixes

- Regressions in certain ProRes decodes since transcoder 4.13 (#3866).

### 18.10.3 5.1.4

*2020-06-18*

#### Improvements

- Be able to choose which transcoder to use for an *essence import* job using the `resourceId` parameter.(#4458).
- Improved IMF support (#4427).
- Add 'vidinetJobId' and 'vidinetResourceId' to analyze job metadata (#4401).

#### Bug fixes

- Vidispine server fails to send/create essence version job to Vidinet (#4422).
- Empty thumbnail result when the "version" query parameter is "all" (#4310).
- Metadata not updated after deleting essence version (#4279).

#### Agent fixes

- Import from VSA where path contains whitespace fails (#4407).

### 18.10.4 5.1.3

*2020-05-07*

#### New features

- Filter out users that are being deleted from the result of "GET /user" (#4302).

#### Improvements

- Be able to create projectVersion when the file path has spacing (#4361).
- Be able to set "duration" to 1 minute for pre-signed URLs (#4336).
- Support SQL Query rewriting for MSSQL (#4324).
- Improve the performance of fetching metadata with "defaultValue=true" in a cluster environment. (#4319).
- Closing raw file chunks should be asynchronous (#4238).

#### Bug fixes

- Thumbnails incorrectly removed after essence version deletions (#4344).
- Cannot create collection with schema validation enabled without specifying bogus collection id (#4236).
- Deletion-lock does not show full siteId in systems using long identifiers (#4195).
- PUT request with empty body does not work from JavaScript (#4175).
- SCC import removes spaces around italic markers (#3737).

#### Transcoder fixes

- Overlays are not working properly for YUV422P16 pixel formats (#4286).

#### Agent fixes

- Files with semicolon in path do not import using VSA (#4230).

## 18.10.5 5.1.2

*2020-04-20*

### Improvements

- Improve large metadata updating speed on item and collection (#4309).
- Abort Vidispine job if transfer step throws a runtime exception (#4305).
- Be able to configure the batch size in DeletionLockBufferCruncher. (#4099).

### Bug fixes

- NPE while closing a file if its storage contains a storage method with unsupported scheme (#4334).
- Fix possible NegativeArraySizeException when uploading large files to S3 (#4256).
- LDAP user sync fails with "too many parameters" SQL Error (#4228).
- EXPORT job is incorrectly marked as FINISHED if one of its steps fails due to being DISAPPEARED (#4200).
- Subtitle rendering is broken for all video formats that are not 4:2:0 (#4024).
- In some cases, thumbnails were not deleted after an item had been removed. (#4021).
- Thumbnailbackground settings in TranscodePresetDocument does not take effect (#3909).

### Transcoder fixes

- Deinterlacing provides wrong frame rate (#4258).
- Closed Captions(EIA-608,CEA-708) not found (#4246).
- Incorrect results from CONFORM job on long GOP material (#3549).

## 18.10.6 5.1.1

*2020-03-18*

### Improvements

- Optimize PSD/PSB reading (#4329).
- Prevent user names been resolved many times during a search request (#4289).
- Add a "recursive" flag to AccessControlType (#4287).

### Transcoder improvements

- Be able to create 8 channels of AAC-LC per track in mp4 (#4317).
- Support decoding AAC in MXF (#4158).

### Bug fixes

- Fix possible user privilege escalation since 5.0 (#4316).
- Fails to connect to an ActiveMQ that requires username and password (#4269).
- Fail to create project version (#4217).
- Items in a child collection not inheriting ancestor's deletion-locks (#4202).
- Search item by metadata field group fails when long identifiers are enabled (#4194).

- Handle PUT (write) to APInoauth URIs pointing to vxa:// resources (#4132).
- Incorrect search result if fields inside a NOT operator have boost factors (#4127).
- Content path not working for transient metadata on the "/API/search" endpoint (#4040).

### Transcoder fixes

- Shape deduction returns wrong duration on certain MXF files (#4243).

### Agent fixes

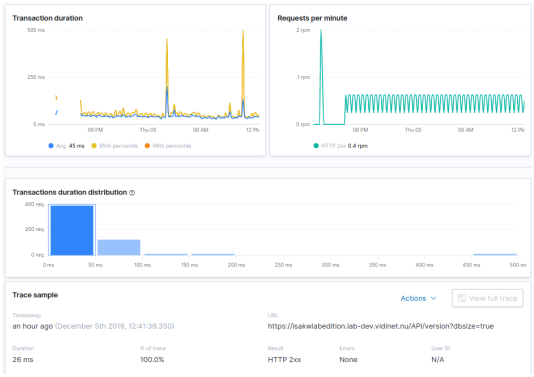
- Files on one VSA storage are incorrectly detected on other VSAs storages (#4273).
- VSA transcode job fails if the original file is in S3 bucket (#4225).
- Can't connect multiple VSAs on the same machine to the same Vidispine server (#4133).

## 18.10.7 5.1

*2020-01-24*

### Elastic APM integration

Vidispine now supports application performance monitoring using *Elastic APM*. It monitors the execution of the application for easy pinpointing of performance issues (#4212).




```
apm:
  elastic:
    urls: ["https://apm.example.com/"]
```

### VSA port forwarding service

Vidispine now supports secure remote *port forwarding using VSA* (#4197). This also enables the possibility to utilize this feature for *LDAP authentication*.

### Preallocated growing files

Local storages can now be configured to `wait for files to be free` before initiating transfers. (#4082). This is useful for Windows storages where the size and metadata is constant while a file is copied to the storage, but the file cannot be read.

### Improvements

- Audit logs can now be configured to `include request bodies and response codes` (#4154).
- It is now possible to filter by priority when *listing jobs* (#4138).
- Allow setting a `priority for storage rule jobs` (#4216).
- Users can now be *created* in a disabled state (#3928).
- Search for *users by group* (#3971).

- Closing raw file chunks is now asynchronous (#4238).
- Avoid segment reading/writing for segments larger than S3 partsize (#4237).
- Include entity id in deletion lock notification (#4078).
- Enable millisecond support for MySQL (#3570).

#### Bug fixes

- Missing allowed values for dataset nodes without parent (#4106).
- Content range in response header longer than complete length if requesting range outside of content length (#4055).
- List item jobs with multiple fields doesn't product the right output (#4111).
- copyFileJob fails with naming script on files that have not been imported. (#4157).
- Error thrown when reading DatabaseSizeLimit property of license (#4112).

#### Upgrading from 5.0

- APIinit is needed due to a new job step for raw and essence imports.
- Solr: No changes to the documents. Re-indexing is not required.

## 18.11 Version 5.0

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.11.1 5.0.9

*2020-10-20*

#### Bug fixes

- User can assign himself to a group without _group_write permission (#4546).
- Imports failing due to Infinispan lock errors for XMP fields (#4487).
- Bulk collection delete fails if collection has ancestor relationship (#4405).
- Direct S3 to S3 transfers not used when useSegmentFiles is true (#4383).
- Metadata modified notification triggered on empty changes (#4465).
- Renamed file not re-detected until after 5 hours (#4579).
- No write from transcoder in 1 hour causes mutable range write to fail (#4575).
- Error retrieving thumbnails from VSA (#4165).

### 18.11.2 5.0.8

*2020-08-31*

### Bug fixes

- Prevent possible JGroups "split-brain" issue during VidiCore startup (#4518).
- Mimetype missing on new shape versions (#4514).
- VidiCore fails to update VidiCoder License in some cases (#4494).
- Incorrect search hit on the */search* endpoint in some cases (#4488).
- Possible OOM when updating metadata referenced in many places (#4484).
- Incorrect timecode if fetching metadata containing "inherited timespan" using the "interval" query parameter (#4368).
- Export with both tag and interval ignores the tag (#4156).
- Removed child collection is still available from parent collection search (#4404)

### Transcoder fixes

- Regressions in certain ProRes decodes since transcoder 4.13 (#3866).

## 18.11.3 5.0.7

*2020-06-18*

### Improvements

- Be able to choose which transcoder to use for an *essence import* job using the `resourceId` parameter.(#4458).
- Improved IMF support (#4427).
- Add 'vidinetJobId' and 'vidinetResourceId' to analyze job metadata (#4401).

### Bug fixes

- Vidispine server fails to send/create essence version job to Vidinet (#4422).
- Empty thumbnail result when the "version" query parameter is "all" (#4310).
- Metadata not updated after deleting essence version (#4279).

### Agent fixes

- Import from VSA where path contains whitespace fails (#4407).

## 18.11.4 5.0.6

*2020-05-06*

### Improvements

- Be able to set "duration" to 1 minute for pre-signed URLs (#4336).
- Support SQL Query rewriting for MSSQL (#4324).
- Improve the performance of fetching metadata with "defaultValue=true" in a cluster environment. (#4319).
- Improve large metadata updating speed on item and collection (#4309).
- Abort Vidispine job if transfer step throws a runtime exception (#4305).
- Be able to configure the batch size in DeletionLockBufferCruncher. (#4099).

### Bug fixes

- Fix possible NegativeArraySizeException when uploading large files to S3 (#4256).
- Cannot create collection with schema validation enabled without specifying bogus collection id (#4236).
- LDAP user sync fails with "too many parameters" SQL Error (#4228).
- EXPORT job is incorrectly marked as FINISHED if one of its steps fails due to being DISAPPEARED (#4200).
- Deletion-lock does not show full siteId in systems using long identifiers (#4195).
- PUT request with empty body does not work from JavaScript (#4175).
- Subtitle rendering is broken for all video formats that are not 4:2:0 (#4024).
- Thumbnailbackground settings in TranscodePresetDocument does not take effect (#3909).
- SCC import removes spaces around italic markers (#3737).

### Transcoder fixes

- Overlays are not working properly for YUV422P16 pixel formats (#4286).
- Deinterlacing provides wrong frame rate (#4258).
- Closed Captions(EIA-608,CEA-708) not found (#4246).

### Agent fixes

- Files with semicolon in path do not import using VSA (#4230).

## 18.11.5 5.0.5

*2020-03-17*

### Improvements

- Prevent user names been resolved many times during a search request (#4289).
- Add a "recursive" flag to AccessControlType (#4287).

### Transcoder improvements

- Be able to create 8 channels of AAC-LC per track in mp4 (#4317).
- Support decoding AAC in MXF (#4158).

### Bug fixes

- Fix possible user privilege escalation since 5.0 (#4316).
- Fails to connect to an ActiveMQ that requires username and password (#4269).
- Fail to create project version (#4217).
- Items in a child collection not inheriting ancestor's deletion-locks (#4202).
- Search item by metadata field group fails when long identifiers are enabled (#4194).
- Handle PUT (write) to APInoauth URIs pointing to vxa:// resources (#4132).
- Incorrect search result if fields inside a NOT operator have boost factors (#4127).
- Content path not working for transient metadata on the "/API/search" endpoint (#4040).

### Transcoder fixes

- Shape deduction returns wrong duration on certain MXF files (#4243).

### Agent fixes

- VSA transcode job fails if the original file is in S3 bucket (#4225).
- Can't connect multiple VSAs on the same machine to the same Vidispine server (#4133).

## 18.11.6 5.0.4

*2020-01-21*

### Bug fixes

- Get metadata's allowed-value with large dataset is slow (#4185).
- Rename operations should not delete source or destination file on failure (#4233).
- Search on user groups is inconsistent (#3931).
- NPE in boost search with wrong sorting field (#4125).

### Transcoder fixes

- Shape deduction of SVG fail unless rsvg-convert is installed (#4199).

## 18.11.7 5.0.3

*2019-12-20*

### Bug fixes

- PUT APIinit/schema/update fails for Solr as the /solr/admin/system endpoint has moved (#4166).
- Elasticsearch health check fails with scope not active error (#4108).
- Dataset values not returned in metadata on item search (#4103).
- Item bulk delete endpoint does not verify deletion locks (#4130).
- SyntaxError while searching with AND OR NOT in text (#4003).
- Boolean fields with capital letters do not index properly in Elasticsearch (#4110).
- Missing administrator role check for the Vidispine log endpoints (#4140).
- VSA SSH key exposed in configuration properties (#4139).
- Users authenticated via external Shiro realm are automatically enabled (#4129).
- Global metadata group/field update without UUID fails with ambiguous update error (#4137).
- Shape is disconnected from item before shape deletion notification is triggered (#4074).
- Can not update "group" and "contentType" in notification action (#4080).
- Storage id cannot be updated in file notification (#3938).

### Transcoder fixes

- 24- and 32-bit audio not handled by Nablet AAC encoder (#4232).

## 18.11.8 5.0.2

*2019-11-19*

### Bug fixes

- UUID overwritten when updating global metadata (#4109).
- Missing allowed values for dataset nodes without parent (#4106).
- Shape content missing from item list output (#3930).

### Transcoder fixes

- Redundant ADIF header in MOV/MP4 output with Nablet AAC (#4148).
- Transcoder crashing on extensive MediaInfo metadata (#4134).

## 18.11.9 5.0.1

*2019-10-31*

### Improvements

- Enable support to explicitly move files with an inherited deletion-lock (#3976).
- Support search for values on dataset fields (#3709).

### Bug fixes

- Regression in 4.17.0, performance issue while building ACL cache (#4104).
- API/whoami returns raw SSO ID, and not the username of the logged in user (#4102).
- SQL syntax error for document metadata indexing when using MySQL (#4096).
- Regression in 5.0.0, Solr self test fails (#4092).
- Incorrect job data when calling job.setData with an array (#4090).
- Shiro user sync failing if user already exists with alias (#4065).
- NPE when updating a storage using StorageDocument with empty sequence tag (#4054).
- Item deletion fails with thumbnails on Azure blob storage (#3979).
- Projects don't inherit deletion-lock (#3944).
- ignoreSidecarImport=true not adhered during placeholder import (#3934).
- Metadata ACL doesn't work if the field is in a group (#3918).
- Scheduled requests not honoring the apiUri configuration property (#3911).
- Import of IMF package failed (#3894).

### Transcoder fixes

- Transcoder is using a visually inferior interpolation method (#4071).
- Transcoder fail to set mediainfo on last audio stream when timecode stream has id 0 (MXF) (#3972).

### Agent fixes

- Regression in 5.0.0, Agent depends on old jaxb2-basics-runtime library (#4100).

## 18.11.10 5.0

*2019-10-02*

### Transcoding using Bitmovin

Vidispine now supports *transcoding using Bitmovin* on Vidinet. (#3820) This is supported for files on S3.

### VSA to VSA transfers

Vidispine Server Agent now supports *direct transfers between VSAs* (#3582). By configuring several VSAs with the same `agentGroup` they can directly transfer files between each other instead of using Vidispine Server as a middle man.

Example of agent configuration:

```
<VXADocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <uuid>aa4a7ef6-087c-4003-82fb-983c0e91d9c3</uuid>
    <name>Test agent</name>
    <uri>http://localhost:57893/</uri>
    <agentGroup>office-vsa-group</agentGroup>
    ...
</VXADocument>
```

### New AAC encoder

A new AAC encoder library from Nablet has been implemented (#3799). This now adds support for creating AAC files with up to 8 channels. To enable the new codec set the codec element to `nablet_aac`. This new codec also requires an updated license key.

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mp4</format>
  <audio>
    <codec>nablet_aac</codec>
    <bitrate>128000</bitrate>
  </audio>
  <video>
    <codec>h264</codec>
    <bitrate>2000000</bitrate>
  </video>
</TranscodePresetDocument>
```

### New XDCamHD encoder

A new XDCamHD encoder library from Nablet has been implemented (#3741). To enable the new codec set the codec element to `nablet_mpeg2video` or `mpeg2video`. This new codec also requires an updated license key. If your license allows for both Nablet and MainConcept XDCamHD, the codec tag `mpeg2video` will default to Nablet version (if the preset is an XDCamHD valid preset). To explicitly use MainConcept, you can use the old codec tag `mc_mpeg2video`.

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mov</format>
  <audio>
    <codec>pcm_s16le</codec>
  </audio>
  <video>
    <codec>nablet_mpeg2video</codec>
    <preset>xdcam_hd_420_1280</preset>
```

```
  </video>
</TranscodePresetDocument>
```

### New H.264 encoder

A new H.264 encoder library from Nablet has been implemented (#3427). To enable the new codec set the codec element to `nablet_h264` or `h264`. This new codec also requires an updated license key. If your license allows for both Nablet and MainConcept H.264, the codec tag *h264* will default to Nablet version. To explicitly use MainConcept, you can use the new codec tag `mc_h64`.

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>mov</format>
  <audio>
    <codec>aac</codec>
    <bitrate>128000</bitrate>
  </audio>
  <video>
    <codec>nablet_h264</codec>
    <bitrate>2000000</bitrate>
  </video>
</TranscodePresetDocument>
```

### JavaScript execution using GraalVM

Vidispine now uses GraalVM JavaScript (https://github.com/graalvm/graaljs) as the default *JavaScript engine*. GraalVM is ECMAScript 2018 compliant. What engine to use can be set on per script basis or as a system-wide setting (#3898).

Example using default parameters and object literals:

```
function getDetail(item, tag='original') {
  let title = "...";
  let created = "...";
  return {
   title, created
  };
}
```

### Searching for documents

Document metadata can now be searched for using the *document search endpoint*. Note that document indexing is disabled by default. Enable by setting *`indexDocumentMetadata`* to `true`.

```
PUT /document
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine" version="2">
  <text>Sweden</text>
</ItemSearchDocument>
```

Existing documents can be indexed using a *reindex request*.

### Simplified migrations

Users no longer need to run both `db init` and `db migrate` to ensure an up to date database schema. The only command needed is `db migrate`. Any init commands will now redirect to this new all-inclusive migration mechanism. We recommend users to update their process to only call `db migrate`.

### Vidispine is now Spring-powered

Vidispine Server no longer uses OpenEJB but has migrated to Spring. As a user, this does not change the way Vidispine works. However, this means that *EJB notifications* and EJB task definitions are no longer supported. Plain Java classes or Spring beans can be used instead.

### Support for matrix parameters has been deprecated

Vidispine no longer uses matrix parameters for requests made to the server. They have been replaced by query parameters of the same name. Note that your current requests using matrix parameters will function for now, but may fail in later versions of Vidispine. We urge all users to update their requests to only use query parameters from this point on. Support for matrix parameters can be disabled using the *convertMatrixParameters* setting.

Previous usage of matrix parameters:

```
;paramX=x;paramY=y
```

New usage of query parameters:

```
?paramX=x&paramY=y
```

### Java 11

Vidispine server and Vidispine server agent (VSA) now use Java 11.

### Solr

Supported Solr version has been changed to 8.1.0 (#3832)

### Elasticsearch

Supported Elasticsearch version has been changed to 6.8.1 (#3688)

### New features

- New endpoint to send job metadata in body when starting a custom job (#3598).
- Allow *absolute timecodes* of timed metadata to be inherited from collections to items (#3550).
- Add a _user_read role that grants read access to user information (#3458).
- Support for *S3 Storage SNS* (Simple Notification Service) (#3265).
- New role _item_write required to *delete an item* (#981).

### Improvements

- Transcoder should give proper error message in case of missing license (#3996).
- Support mutable writes directly to Google Cloud Storage (#3984).
- Support creation of signed URLs for Google Cloud Storage (#3982).
- Allow Google Cloud Storage private key to be embedded in URL (#3980).
- Speed up the fetching of thumbnails from no-auth URL (#3958).
- Make it possible to *control which tracks are written* in CPL and physically based on shape (#3861).
- New license parameter to limit database size (#3844).
- Support decoding of “qtrle” for all licenses (#3826).

- Include cluster in selftest documentation (#3778).
- Support *adding external hash values* to files (#3610).
- Speed up Vidinet jobs by requesting a media check of output file(s) in the transcode job (#3603).
- Support new AWS regions even though they are not in the included SDK (#2731).

### New transcoder features

- Support for *extracting alpha layer* and encode as a separate output (#3876).

### Transcoder improvements

- Speed up PSD/PSB to low-res (#3772).

### Bug fixes

- Can not tell which metadata-field caused an error on update (#3950).
- S3 bucket online check fails when lacking S3:ListBucket permission (#3914).
- COPY_FILE job does not copy the entire file when the source file is growing (#3882).
- Python SDK handling of binary data (#3851).
- Users can not read their own metadata via the metadata endpoint (#3843).
- Missing element, expire, in deletion-lock notification (#3803).
- OPTIONS results in 403 Forbidden response for endpoint collection/{id}/item (#3801).
- Terse as JSON produce broken output when using Jackson (#3740).
- Thumbnail in item metadata refer to sub layer, instead of flattened image for multi-layered files (#3714).
- URI generation tool from APIdoc generates bad URI for special characters (#3574).

### Transcoder fixes

- Can not create 96kHz audio tracks in MP4 container (#3978).
- A/V out-of-sync when creating sub clips with no transcode (#3961).

### Upgrading from 4.17

> **Warning:** Because of a change in our internal system for the handling of database migrations, all users **MUST** upgrade to version 4.17 of Vidispine server and complete the database migration process before attempting to upgrade to Vidispine Server 5.0.

- Reindex needed due to Solr/Elasticsearch upgrade and changes to how dataset field values are indexed.
- APIinit is needed due to new roles being added.
- Solr: Supported version has been changed to 8.1.0.
- Elasticsearch: Supported version has been changed to 6.8.1.
- ActiveMQ: Supported version has been changed to 5.15.9.

For more information on upgrading to 5.0, see *Upgrading to Vidispine 5.0*.

## 18.12 Version 4.17

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.12.1 4.17.10

*2020-10-19*

#### Bug fixes

- Renamed file not re-detected until after 5 hours (#4579).
- No write from transcoder in 1 hour causes mutable range write to fail (#4575).
- Error retrieving thumbnails from VSA (#4165).

### 18.12.2 4.17.9

*2020-08-31*

#### Bug fixes

- Mimetype missing on new shape versions (#4514).
- Possible OOM when updating metadata referenced in many places (#4484).
- Export with both tag and interval ignores the tag (#4156).

#### Transcoder fixes

- Regressions in certain ProRes decodes since transcoder 4.13 (#3866).

### 18.12.3 4.17.8

*2020-06-08*

#### Improvements

- Improved IMF support (#4427).

#### Bug fixes

- Empty thumbnail result when the "version" query parameter is "all" (#4310).
- Metadata not updated after deleting essence version (#4279).

#### Transcoder fixes

- Shape deduction of SVG fail unless rsvg-convert is installed (#4199).

#### Agent fixes

- Import from VSA where path contains whitespace fails (#4407).

### 18.12.4 4.17.7

*2020-05-05*

### Improvements

- Be able to set "duration" to 1 minute for pre-signed URLs (#4336).
- Improve large metadata updating speed on item and collection (#4309).
- Abort Vidispine job if transfer step throws a runtime exception (#4305).
- Add a "recursive" flag to AccessControlType (#4287).
- Be able to configure the batch size in DeletionLockBufferCruncher. (#4099).

### Bug fixes

- Incorrect parsing of SCC timecodes (#4314).
- NPE when timeout occurs (#4284).
- Fix possible NegativeArraySizeException when uploading large files to S3 (#4256).
- LDAP user sync fails with "too many parameters" SQL Error (#4228).
- EXPORT job is incorrectly marked as FINISHED if one of its steps fails due to being DISAPPEARED (#4200).
- HEAD request to thumbnail URL returns Content-Length: 0 (#4188).
- PUT request with empty body does not work from JavaScript (#4175).
- Incorrect search result if fields inside a NOT operator have boost factors (#4127).
- Thumbnailbackground settings in TranscodePresetDocument does not take effect (#3909).
- SCC import removes spaces around italic markers (#3737).

### Transcoder fixes

- Overlays are not working properly for YUV422P16 pixel formats (#4286).
- Deinterlacing provides wrong frame rate (#4258).
- Closed Captions(EIA-608,CEA-708) not found (#4246).
- Shape deduction returns wrong duration on certain MXF files (#4243).

### Agent fixes

- 4.17.4 agent fails to start (#4257).
- Files with semicolon in path do not import using VSA (#4230).
- VSA transcode job fails if the original file is in S3 bucket (#4225).

## 18.12.5 4.17.6

*2020-01-21*

### Bug fixes

- Get metadata's allowed-value with large dataset is slow (#4185).
- Rename operations should not delete source or destination file on failure (#4233).
- Search on user groups is inconsistent (#3931).
- NPE in boost search with wrong sorting field (#4125).

### 18.12.6 4.17.5

*2019-12-20*

#### Bug fixes

- Dataset values not returned in metadata on item search (#4103).
- Item bulk delete endpoint does not verify deletion locks (#4130).
- SyntaxError while searching with AND OR NOT in text (#4003).
- Boolean fields with capital letters do not index properly in Elasticsearch (#4110).
- Missing administrator role check for the Vidispine log endpoints (#4140).
- VSA SSH key exposed in configuration properties (#4139).
- Users authenticated via external Shiro realm are automatically enabled (#4129).
- Global metadata group/field update without UUID fails with ambiguous update error (#4137).
- Shape is disconnected from item before shape deletion notification is triggered (#4074).
- Can not update “group” and “contentType” in notification action (#4080).
- Storage id cannot be updated in file notification (#3938).

### 18.12.7 4.17.4

*2019-11-19*

#### Bug fixes

- UUID overwritten when updating global metadata (#4109).
- Missing allowed values for dataset nodes without parent (#4106).
- Shape content missing from item list output (#3930).

#### Transcoder fixes

- Transcoder is using a visually inferior interpolation method (#4071).

### 18.12.8 4.17.3

*2019-10-25*

#### Improvements

- Enable support to explicitly move files with an inherited deletion-lock (#3976).
- Support search for values on dataset fields (#3709).

#### Bug fixes

- Regression in 4.17.0, performance issue while building ACL cache (#4104).
- API/whoami returns raw SSO ID, and not the username of the logged in user (#4102).
- Shiro user sync failing if user already exists with alias (#4065).
- NPE when updating sequence in storage (#4054).
- Item deletion fails with thumbnails on Azure blob storage (#3979).

- Project collection cannot inherit deletion-lock (#3944).
- Metadata ACL doesn't work if the field is in a group (#3918).
- Scheduled requests not honoring the apiUri configuration property (#3911).
- Importing IMF package failed (#3894).

### Transcoder fixes

- Transcoder is using a visually inferior interpolation method (#4071).
- Transcoder fail to set mediainfo on last audio stream when timecode stream has id 0 (MXF) (#3972).

## 18.12.9 4.17.2

*2019-09-20*

### New features

- Allow analyze jobs to run as part of a transcode/import job (#3867).

### Improvements

- Enable metadata update requests to *only return changed entries* (#3974).
- Allow collection to be *retrieved without items, collections or libraries* (#3927).
- Support Google Cloud transcoders in Vidinet (#3959).
- Improve performance of global metadata updates for large documents (#3965).
- Improve performance of metadata group search on global metadata (#3946).
- Faster removal of global metadata by UUID (#3922).
- Expose creation time for shapes (#3899).
- Support creation of shapes with components referencing different files (#3977).
- Allow user aliases to be set via Shiro (#3960).
- Reduce possible database locking error on "t_file" (#3923).
- Add flag to limit import of unfulfilled partial IMPs (#3853).

### Bug fixes

- Metadata group is still returned after deletion in document metadata (#3917).
- Missing boost factor for phrase query in Solr (#3901).
- Components can get incorrect itemTrack when using path projections (#3895).
- Incorrect search result if query contains a "NOT" operator and a boost factor (#3878).
- Dataset field validation to support multiple hierarchies (#3994).
- Missing recursive element in item ACL response (#3887).
- VSA should only close proxy URIs that VSA has created, not proxy URIs that VS has created (#3892).
- VXA URIs should always be accepted for VSA transcoders with the same host (#3862).
- Use IMF InsintricDuration if SourceDuration is not present (#3849).

### Transcoder fixes

- Watermark/timecode appearing in formats that should not have watermark (#3933).

## 18.12.10 4.17.1

*2019-06-28*

### New transcoder features

- Support for extracting IPTC-metadata from images (#3710).

### Bug fixes

- Job step marked as DISAPPEARED even though sub-steps are running (#3865).
- Render job fails if sequence does not contain tracks for all audio channels (#3839).
- NPE on LDAP sync if name attribute is missing from a user entry (#3838).
- Indexing failing for item with thousands of files (#3837).
- TTML timecode parser does not handle fractions (#3831).
- Incorrect essence version on item if essence import job fails (#3830).
- Binary components missing from shape after placeholder import of binary component (#3828).
- CloudConvert script not evaluated for essence import jobs (#3827).
- File on StorNext archive not shown as ARCHIVED (#3824).
- File on Glacier Deep Archive not shown as ARCHIVED (#3823).
- Placeholder import job incorrectly overwrites original shape metadata (#3821).
- Transcode of document formats not working with CloudConvert (#3817).
- Scheduled requests not working for all endpoints (#3815).
- Incorrect username migration query shown on dry-run (#3787).
- Import settings not applied during import to placeholder item (#3770).
- Normalizer class cannot be accessed from sandboxed JavaScript (#3749).
- Retrieving storages by storage group doesn't work since 4.16 (#3748).
- Dataset values not displayed in search results (#3734).
- Shape import of .mov file creates item with wrong mimeType and mediaType (#3732).
- Unable to update all action fields on a notification (#3705).
- Unable to update trigger on notifications (#3704).
- EJB notification is not fired if synchronous is set to false (#3703).
- Items and collections not reindexed after an external idenfitier change (#3697).
- Can't import to placeholder that was created before upgrade to 4.16.1 (#3693).
- Too large document_text causing too large messages on the index queue (#3669).
- Integer values accepted as input for date fields (#3665).
- Shape import of IMF package does not set system metadata, like original codec etc. (#3301).

### Transcoder fixes

- Transcoder generates invalid MXF (#3889).
- Slow media check of tar file on S3 storage (#3835).
- Incorrect bit rate metadata for MPEG transport stream (#3788).
- Burn in timecode without background bar causes text shadow (#3777).
- Shape deduction of SVG files take a very long time (#3720).

### Agent fixes

- Shape deduction by agent fails for file on S3 with special characters (#3563).

## 18.12.11 4.17

*2019-03-29*

### PostgreSQL 11 and Ubuntu 18.04

PostgreSQL 11 and Ubuntu 18.04 are now officially supported.

### Configure OAuth authentication using API

The *OAuth* configuration can now be set using the new *auth configuration endpoint*. Previously it was supported by adding the configuration in the Shiro configuration file.

See *Configure OAuth2 using the API* for more details.

### Trigger notifications from JavaScript

Notifications can now be sent from JavaScript job steps (#3454). This is done using the *notification object*. This works for all notification actions.

For example, given an SQS notification with id VX-45, adding a message to that queue could be done using:

```
notification.send("VX-45", {
   "item": "VX-1",
   "state": "VALID"
});
```

### Import of TTML subtitle sidecar files

Import and basic parsing of TTML (https://www.w3.org/TR/2018/REC-ttml1-20181108) subtitle files is now supported (#3581). The content of TTML files will be parsed as *subtitle metadata* on the items. Currently supported fields are `stl_text`, `stl_justification`, `stl_color` and `stl_font`.

### Storage priority

The user can now assign a *priority to a storage*. When a source file exists on multiple storages Vidispine server will use the file from the storage with the highest priority. This is, for example, useful if one storage has much faster retrieval speed than the other(s) (#3578).

### Control if a storage is active/inactive

This feature allows the user to mark a storage as active by adding a .storage file in the root of the storage. To enable this check for all storages you can add *`storageActivationFile`* to the server.yaml. To only enable it for a specific storage you can set the key *`storageActivationFile`* in the storage metadata (#3552).

### Improved status of thumbnail resources

The thumbnail resource has been updated to contain online/offline information (#3228).

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-6</id>
  <thumbnail>
    <path>file:///srv/thumbnails/</path>
    <state>ONLINE</state>
    <lastSuccess>2019-03-01T10:42:55.239+01:00</lastSuccess>
  </thumbnail>
</ResourceDocument>
```

### Improved access control options

A new `appliesTo` setting is introduced for finer control of the entities that an *access control* entry is inherited to. This replaces the previous `recursive` option (#2369).

### Improvements

- Preserve access granted by disabled users (#3631).
- Support for aborting jobs in Vidinet (#3630).
- *Metadata locks* on collections (#3590).
- Retry to send job to Vidinet if Vidinet is temporarily inaccessible instead of failing the Vidispine job (#3528).
- Support for using *Nexio FTP-servers* as storages (#3514).
- Skip unnecessary media checks for files on object storages (#3511).
- New endpoint to *set multiple configuration properties* at once (#3451).
- Support to *get all key/value metadata* of metadata-field/group (#3438).
- Validate bucket region when creating an S3 storage (#3147).

### Bug fixes

- IOException for FTP storage when pooling is enabled (#3700).
- Get metadata not accepting all timecode formats of interval parameter (#3657).
- HTTP 500 when adding access controls in bulk (with -Xss256k) (#3617).
- Stack overflow error when running with a smaller stack size (#3612).
- High memory usage due to a large number of OpenEJB scheduler threads (#3611).
- Error message in log if no S3 bucket configured for audit trail purging (#3584).
- Thumbnail job does not honor resolution set in shape-tag (#3579).
- Search operator name not validated when using Elasticsearch (#3567).
- Cannot import sidecar with fileId on Vidinet (#3558).
- Jobs that have been in WAIT state cannot be deleted (#3555).
- Bad performance of collection/{cid}/graph/dot if ancestor collections have a lot of items (#3546).
- Permissions created by grantor persist after deleting grantor account (#3535).
- URI-escaping of files detected on Azure storages is not working properly (#3532).

- markDeletedFilesMissing gets stuck on huge amount of entries (#3428).
- Metadata changes on inherited metadata are not reflected in search index (#3238).

#### Transcoder fixes

- Improve transcoder retry logic for mutable regions (#3692).

#### Agent fixes

- Allow moving files within the same VSA storage (#3426).
- VSA rewrites CIFS URLs (#3391).

#### Platform

This release adds support for Ubuntu 18.04 and PostgreSQL 11. At the same time, support for PostgreSQL 9.1 and 9.3, MySQL 5.5 and Ubuntu 14.04 has been discontinued.

#### Upgrading from 4.16

- APIinit is needed due to new metadata fields added for TTML parsing.
- Solr: No changes to the documents. Re-indexing is not required.

## 18.13 Version 4.16

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.13.1 4.16.7

*2019-12-18*

#### Bug fixes

- Global metadata group/field update without UUID fails with ambiguous update error (#4137).
- Shape is disconnected from item before shape deletion notification is triggered (#4074).
- Can not update "group" and "contentType" in notification action (#4080).
- Storage id cannot be updated in file notification (#3938).

#### Transcoder fixes

- Transcoder is using a visually inferior interpolation method (#4071).

### 18.13.2 4.16.6

*2019-09-20*

#### New features

- Allow analyze jobs to run as part of a transcode/import job (#3867).

### Improvements

- Enable metadata update requests to *only return changed entries* (#3974).
- Allow collection to be *retrieved without items, collections or libraries* (#3927).
- Support Google Cloud transcoders in Vidinet (#3959).
- Improve performance of global metadata updates for large documents (#3965).
- Improve performance of metadata group search on global metadata (#3946).
- Faster removal of global metadata by UUID (#3922).
- Expose creation time for shapes (#3899).

### Bug fixes

- Metadata group is still returned after deletion in document metadata (#3917).
- Missing boost factor for phrase query in Solr (#3901).
- Components can get incorrect itemTrack when using path projections (#3895).
- Incorrect search result if query contains a "NOT" operator and a boost factor (#3878).

## 18.13.3 4.16.5

*2019-07-03*

### Bug fixes

- Retrieving storages by storage group doesn't work since 4.16 (#3748).

## 18.13.4 4.16.4

*2019-06-28*

### Bug fixes

- Indexing failing for item with thousands of files (#3837).
- Binary components missing from shape after placeholder import of binary component (#3828).
- CloudConvert script not evaluated for essence import jobs (#3827).
- File on StorNext archive not shown as ARCHIVED (#3824).
- File on Glacier Deep Archive not shown as ARCHIVED (#3823).
- Placeholder import job incorrectly overwrites original shape metadata (#3821).
- Incorrect username migration query shown on dry-run (#3787).
- Dataset values not displayed in search results (#3734).
- Unable to update all action fields on a notification (#3705).
- Unable to update trigger on notifications (#3704).
- EJB notification is not fired if synchronous is set to false (#3703).
- Items and collections not reindexed after an external idenfitier change (#3697).
- Can't import to placeholder that was created before upgrade to 4.16.1 (#3693).
- Too large document_text causing too large messages on the index queue (#3669).

- Integer values accepted as input for date fields (#3665).

### Transcoder fixes

- Transcoder generates invalid MXF (#3889).
- Slow media check of tar file on S3 storage (#3835).
- Shape deduction of SVG files take a very long time (#3720).

### Agent fixes

- Shape deduction by agent fails for file on S3 with special characters (#3563).

## 18.13.5 4.16.3

*2019-03-28*

### Bug fixes

- Mutable range write failing if transcoder writes large "gaps" in file (#3713).
- Move metadata migration fails if target group exists (#3708).
- Very slow file write completion when using segment files on Azure (#3694).
- Failing to reindex collection if the collection number is high (#3668).
- Incorrect normalization of combining characters for JSON (#3663).
- Filename scripts that fail due to a temporary error causes jobs to fail instead of retry (#3573).
- Transcoding documents using CloudConvert fails (#3533).

### Transcoder fixes

- MPEG-TS erroneously detected as MPEG-PS (#3666).
- Closed captions not recognized in MXF (#3620).
- Transcoder hangs on media check of tar file with encrypted MXF (#3559).
- Automatically adjust odd dimensions for H.264 output (#3414).

## 18.13.6 4.16.2

*2019-03-13*

### Bug fixes

- Slow add/delete of item from large collection (#3671).
- CORS does not work with /API/user/{user-id} (#3662).
- Transfers to storage with read=false on method end up in WAITING state (NO_SUCH_METHOD) (#3622).
- Signed S3 URL generation doesn't take into account SSE algorithm configuration (#3619).
- Jobs that had been in WAIT state can not be deleted (#3555).
- Unable to use scene change detection with thumbnail jobs (#3493).
- Repeated "Ingoring no longer waiting job" error in logs (#3681).

### Transcoder fixes

- Transcode crashing on transcode of certain PSD files (#3691).

### Agent fixes

- Incorrect Guava version in VSA 4.16.1 causing writes to fail (#3676).
- S3 files are not properly written by VSA when using mutable range writes (#3664).

## 18.13.7 4.16.1

*2019-02-08*

### Improvements

- Resource for listing the *libraries that contain a specific item* (#3547).

### Bug fixes

- Using empty:/// URI does not work for truncating audit log and transfer logs (#3609).
- HTTP 400 when setting external ids or metadata field groups in metadata (#3599).
- Incorrect file list counts if a file belongs to multiple items for Elasticsearch (#3595).
- Incorrect search result if the search value contains "literal" spaces for Elasticsearch (#3594).
- Auto-completion not case insensitive for string-exact for Elasticsearch (#3593).
- SAML authentication broken with Bouncy Castle 1.60 (#3588).
- Too strong SSH key generated by Vidispine on Java 1.8u91 and later (#3583).
- Proxy links for manually getting file data does not respect 'apiNoauthUri' property (#3096).
- Sorting users by username/realname not working when searching for users (#2961).

### Transcoder fixes

- Tearing in video output when deinterlacing source (#3606).

## 18.13.8 4.16

*2019-01-13*

### Sequence rendering using Vidinet

Sequence rendering can now be performed using Vidinet. This works just like when using a Vidinet transcoder to transcode or conform (#3221).

```
POST /sequence/render?tag=h264&resourceId=VX-42
```

### Stream files from agent

You can now get access to read files directly from the VSA. Previously you could only access the files by streaming them through Vidispine. This make sense, if you for example are on the same on-premise network as the VSA and you have Vidispine server running in the cloud. The access is requested with a new methodType: *AUTO-VSA*.

### Paging using cursors

It is now possible to use the cursorMark (https://lucene.apache.org/solr/guide/6_6/pagination-of-results.html#fetching-a-large-number-of-sorted-results-cursors)/search after (https://www.elastic.co/guide/en/elasticsearch/reference/6.3/search-request-search-after.html#search-request-search-after) features from Solr/Elasticsearch to improve the deep paging (https://lucene.apache.org/solr/guide/6_6/pagination-of-results.html#performance-problems-with-deep-paging) performance during a search (#3476).

This feature is supported on *collection* and *item* search as well as when *listing files* using the new the `cursor` parameter.

---

**Note:** Only metadata searches in the `generic` interval supports `cursor`.

---

### Query boosting

It is now possible to *boost* field values in item and collection searches by adding boost factors in the search documents or metadata field definitions.

```
PUT /item
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine" version="2">
  <operator operation="OR">
    <field>
      <name>title_field</name>
      <value boost="10">phoenix</value>
    </field>
    <field>
      <name>description_field</name>
      <value>phoenix</value>
    </field>
  </operator>
  <sort>
    <name>_score</name>
    <value>descending</value>
  </sort>
</ItemSearchDocument>
```

```
<MetadataFieldDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <name>title_field</name>
  <type>string</type>
  <boost>10.0</boost>
</MetadataFieldDocument>
```

### Key-value metadata improvements

Support for *key-value metadata* has been added to:

- Shape tags (#3440)
- Libraries (#3412)
- Task definitions (#3439)

The JavaScript job object has also been updated with additional functions for working with the job data (#3441 and #3437):

- *job.getKeys()*
- *job.containsKey()*
- *job.getDataOrDefault()*

### Improvements

- Support for *bulk upload* of files to a placeholder shape (#3434).
- Option to configure which file extensions to *not treat as sidecar files* (#3430).
- Added missing permissions for multipart upload using *temporary S3 credentials* (#3548).
- Support for retrieving *temporary S3 credentials* from the EC2 instance profile (#3490).
- Ability to filter on groups that are roles when *searching for groups* (#3487).
- Support for pagination when *searching for users* (#3273).
- Support for creating *multiple (item) relationships* in a single request (#3452).
- Support for disabling the built in SSH server (#3448).
- Support for specifying destination storage of output file for *transcode jobs* (#3424).
- Support for metadata *dataset models in JSON-LD format* (#3416).
- Proper error message if transcoder discovery settings is not correct (#3483).
- Reduce the possible "Lock wait timeout" when running jobs (#3370).
- SMB 2 support (#3510).

### Bug fixes

- Authentication failure when VSA connects over SSH to Vidispine running on Java 1.8u91 or later (#3569).
- Generating a pre-signed Azure URL results in a HTTP URL, expected HTTPS (#3551).
- Only sidecar files with lower case file extensions are picked up as sidecars (#3429).
- Resolution and aspect ratio not marshalled properly with useJackson=false when getting shapes as JSON (#3422).
- OptimisticLockException if JavaScript job problem is checked while job is running (#3565).
- JavaScript notification actions cannot be configured when running on a MS SQL Server database (#3315).
- GET or DELETE of a non-existent metadata lock returns 500 (#3303).
- Possible metadata migration failure due to long running database queries (#3280).
- Component import to shape fails because of duplicate job creation (#3278).
- Don't allow transient metadata fields to be modified (#3277).
- HTTP 500 response when importing a Final Cut XML that has a project element with no children (#3226).
- Missing mediainfo section for shapes imported by placeholder imports (#3206).
- GET storage ignores the status parameter (#3148).
- NPE instead of proper error message if export template fails to evaluate (#2889).
- Incorrect file key logged for API requests (#3568).

#### Transcoder fixes

- The transcoders StatsD metrics not compliant with collectd (#3192).

#### Upgrading from 4.15

- Solr: No changes to the documents. Re-indexing is not required.

## 18.14 Version 4.15

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.14.1 4.15.4

*2019-06-28*

#### Bug fixes

- CloudConvert script not evaluated for essence import jobs (#3827).
- File on StorNext archive not shown as ARCHIVED (#3824).
- File on Glacier Deep Archive not shown as ARCHIVED (#3823).
- Can't import to placeholder that was created before upgrade to 4.16.1 (#3693).
- Too large document_text causing too large messages on the index queue (#3669).

#### Transcoder fixes

- Slow media check of tar file on S3 storage (#3835).

#### Agent fixes

- Shape deduction by agent fails for file on S3 with special characters (#3563).

### 18.14.2 4.15.3

*2019-03-28*

#### Bug fixes

- Move metadata migration fails if target group exists (#3708).
- Failing to reindex collection if the collection number is high (#3668).
- Incorrect normalization of combining characters for JSON (#3663).
- Filename scripts that fail due to a temporary error causes jobs to fail instead of retry (#3573).
- Transcoding documents using CloudConvert fails (#3533).

#### Transcoder fixes

- MPEG-TS erroneously detected as MPEG-PS (#3666).
- Closed captions not recognized in MXF (#3620).
- Transcoder hangs on media check of tar file with encrypted MXF (#3559).
- Automatically adjust odd dimensions for H.264 output (#3414).

### 18.14.3 4.15.2

*2019-01-11*

#### Bug fixes

- Slow update of Vidinet job status (#3507).
- db migrate from version 4.14.1 to 4.15 fails on MS SQL Server (#3503).
- Content-Disposition header support missing for schemes: file, smb, azure (#3496).
- Relative date search with Elasticsearch gives HTTP 500 (#3489).
- Shape import to placeholder does not set mimeType on item (#3475).

#### Transcoder fixes

- Transcode job fails for .tga file during raw import (#3557).
- Transcoder crash if MP3 encoder receives unsupported input, track with more than 2 channels (#3504).
- Transcoder crashes with unsupported TrueHD input (#3497).

### 18.14.4 4.15.1

*2018-11-12*

#### Bug fixes

- Shape metadata not saved when creating a placeholder shape (#3410).
- Transcoder DNS discovery not using hostname in TLS session (#3468).
- Faceted search with transient fields returns an incorrect field name against Solr (#3402).
- Display values for metadata dataset fields are not returned in /API/item/VX-NN?content=metadata (#3234).
- API/import/imf not honouring Vidinet transcoder (#3484).
- All files get the same componentOriginalFilename when importing to placeholder item in bulk (#3411).
- Metadata field originalFilename not set when using raw placeholder import (#3469).
- Sequence render request ignores destinationItem parameter (#2638).
- Deletion lock incorrectly prevent storage cleanup or file removal (#3466).
- Multiple items created if storage exception is thrown when creating entities. (#3404).
- Import failing when using asterisks (*) in filename (#3376).
- URI content parameter ignores the version parameter (#2790).
- File sequence marked as MISSING during long transcode (#3500).
- HTTP 500 Server error for timed out APInoauth URL (#3485).
- Incorrect exit status from vidispine-admin db-apiinit (#3435).
- Collection creation with attached field group gets progressively slower (#3470).
- CloudConvert job failing if external identifier is used (#3392).
- Wildcard text query with uppercase characters failing with Elasticsearch (#3320).
- NPE when transcoding file on FTP (#3478).

- Placeholder imports to the same item fail with optimistic locking errors on clustered systems (#3477).
- Importing to a full storage creates a never ending job (#3181).
- Exporting using URL with query parameters generates a metadata file without .xml extension (#3480).
- FileNotFoundException in server log when using slave licensing (#3408).
- Metadata subgroups remain in search results after item/collection is deleted (#3196).

### Transcoder fixes

- Wrong field order information from ProRes files (#3461).

### Agent fixes

- VSA tries to re-connect to VS behind load balancer too aggressively (#3482).

## 18.14.5 4.15

*2018-10-12*

### AWS Elemental MediaConvert

Transcoding using *AWS Elemental MediaConvert* is now supported via Vidinet (https://www.vidispine.com/vidinet) (#3420) for files on S3.

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-1</id>
  <vidinet>
    <url>vidinet://e1423727-...:...@42a73b4c-8974-4402-a237-17b80bd11350</url>
    <name>My AWS MediaConvert</name>
    <endpoint>https://services.vidinet.net</endpoint>
    <type>ELEMENTAL_MEDIACONVERT</type>
    <state>ONLINE</state>
    <scheme>s3</scheme>
  </vidinet>
</ResourceDocument>
```

To transcode with the new Vidinet resource you need a new shape-tag with the `mediaconvert` tag in it.

```
 <TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <description>
     BROADCAST, XDCAM, MXF, MPEG2 HD422, WAV, 16x9 DAR, 1920x1080p, 23.98 Hz, 50 Mbps␣
↪CBR
   </description>
   <name>
     __mediaconvert_Broadcast_Xdcam_Mxf_Mpeg2_Wav_16x9_1920x1080p_24Hz_50Mbps
   </name>
   <audio/>
   <video/>
   <mediaconvert>
     <outputSetting>
       { "Type": "SYSTEM", "Category": "BROADCAST-XDCAM", ... }
     </outputSetting>
   </mediaconvert>
 </TranscodePresetDocument>
```

### Temporary credentials for S3/Azure

It is now possible to *generate temporary credentials* for S3 (#3389). These credentials are limited to a specific object and expire after a user defined time period.

```
POST /storage/file/VX-56/uri?scheme=s3&write=true
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>s3://ASIA....:...@example/image.jpg?sessionToken=...</uri>
</URIListDocument>
```

If you prefer to work with HTTP, then pre-signed HTTP URLs can be requested instead. This works with both S3 and Azure Blob Storage.

```
POST /storage/file/VX-98/uri?scheme=http&write=true
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>https://example.blob.core.windows.net/image.jpg?sig=...</uri>
</URIListDocument>
```

### CORS

Vidispine can be configured to emit *Cross-Origin Resource Sharing (CORS) heades* (#3299). For example, to configure that web application running on example.com may access the API:

```
<CORSConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <entry>
    <request/>
    <response>
      <allowOrigin>http://example.com</allowOrigin>
    </response>
  </entry>
</CORSConfigurationDocument>
```

### Deletion locks

It's now possible to set deletion locks on collections, items and files, to prevent them from been moved or deleted (#3283).

```
<DeletionLockDocument xmlns="http://xml.vidispine.com/schema/vidispine" >
  <id>VX-1574</id>
   <user>admin</user>
   <expiryTime>2018-10-22T14:27:03.175+02:00</expiryTime>
   <modified>2018-10-12T14:27:03.175+02:00</modified>
   <entityType>Item</entityType>
   <entityId>VX-32140</entityId>>
</DeletionLockDocument>
```

Deletion locks can be *inherited* from parent to child entities, and used in *searches*. They also support *notifications*.

### New features

- Get storage that would be *used for ingest* (#3390).
- Transcoder discovery via *SRV records* for secure HTTPS transcoder URLs (#3388).

### Improvements

- Allow cycles and self-references in metadata hierarchy (#3394).
- Be able to return full component details using `GET /item/(id)/shape/(shape-id)/component/(component-id)` (#3344).
- Configurable *key and trust store* for HTTPS (#3310).
- Support for automatic restore of S3 objects in CONFORM jobs (#3290).
- Include Atempo job id in job metadata (#3263).

### Transcoder improvements

- Remove incorrect file and folder name in shape metadata (#3433).

### Bug fixes

- Old items/files not removed from Solr after reindex (#3432).
- Placeholder raw import for binary files fail due to NPE (#3349).
- Search index not updated after adding/removing item sequences (__sequence_size) (#3346).
- HTTP 401 when fetching components for shape (#3343).
- Jobs gets stuck if defaultTranscoder is not a valid Vidinet resource (#3328).
- Self tests halting on transcode self test failure (#3325).
- HTTP 500 on transcoder resources when schema validation is enabled (#3307).
- HTTP 500 when fetching job and specifying custom timezone (#3225).
- HTTP 404 on import request using shape-tag with comma in name (#3213).
- Can't set user metadata when creating a new user (#2926).
- Incorrect "loc" element for access controls (#2879).
- Not able to set metadata-field values using JSON (#2755).

### Upgrading from 4.14

- Solr: No changes to the documents. Re-indexing is not required.
- The type of job that is created for the *placeholder raw import request* has changed from THUMBNAIL/TRANSCODE to PLACEHOLDER_IMPORT.

## 18.15 Version 4.14

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.15.1 4.14.6

*2019-06-28*

### Bug fixes

- Can't import to placeholder that was created before upgrade to 4.16.1 (#3693).
- Too large document_text causing too large messages on the index queue (#3669).

#### Transcoder fixes

- Slow media check of tar file on S3 storage (#3835).

### 18.15.2 4.14.5

*2019-03-28*

#### Bug fixes

- Move metadata migration fails if target group exists (#3708).
- Failing to reindex collection if the collection number is high (#3668).
- Incorrect normalization of combining characters for JSON (#3663).

#### Transcoder fixes

- MPEG-TS erroneously detected as MPEG-PS (#3666).
- Transcoder hangs on media check of tar file with encrypted MXF (#3559).
- Automatically adjust odd dimensions for H.264 output (#3414).

### 18.15.3 4.14.4

*2019-01-11*

#### Bug fixes

- Content-Disposition header support missing for schemes: file, smb, azure (#3496).
- Relative date search with Elasticsearch gives HTTP 500 (#3489).
- Shape import to placeholder does not set mimeType on item (#3475).

#### Transcoder fixes

- Transcode job fails for .tga file during raw import (#3557).

### 18.15.4 4.14.3

*2018-11-05*

#### Bug fixes

- Collection creation with attached field group gets progressively slower (#3470).
- CloudConvert job failing if external identifier is used (#3392).
- Wildcard text query with uppercase characters failing with Elasticsearch (#3320).
- NPE when transcoding file on FTP (#3478).
- Placeholder imports to the same item fail with optimistic locking errors on clustered systems (#3477).
- Importing to a full storage creates a never ending job (#3181).
- Exporting using URL with query parameters generates a metadata file without .xml extension (#3480).
- FileNotFoundException in server log when using slave licensing (#3408).
- Metadata subgroups remain in search results after item/collection is deleted (#3196).

### Transcoder fixes

- Wrong field order information from ProRes files (#3461).

## 18.15.5 4.14.2

*2018-09-28*

### Improvements

- Support database purging/export for all supported storages (#3383).

### Agent improvements

- Support checking file status on storage after transfer (#3413).
- Exclude $RECYCLE.BIN from VSA indexed directories (#3321).

### Bug fixes

- Newly added job step not being used by new jobs (#3419).
- Job feedback queue growing after running an ARCHIVE job (#3405).
- Vidispine server opening too many UDP sockets when metrics are enabled (#3401).
- Slow deletion of duplicate metadata documents on db migrate (#3400).
- Transcoder may get more jobs than the configured "maxJobs" (#3375).
- Cannot hard delete user with collections or quota rules (#3356).
- Infinispan cache sometimes gets cleared when a node leaves and rejoins the cluster (#3338).
- Modifying multiple metadata entries in one request gives HTTP 400 (#3289).
- File delete job finishing without removing file from storage (permission error) (#3284).
- Out of memory error when migrating username metadata (#3355).
- No thumbnails created on import with transcode disabled (#3353).
- Migration failing on MySQL when multiple databases exists (#3331).
- MySQL migration does not handle primary keys properly on clustered MySQL (#3329).
- Database migration from 4.13 fails on first attempt on MySQL (#3316).
- CloudConvert transcode fails when default transcoder is Vidinet (#3351).
- Media check using incorrect Vidinet transcoder (#3384).
- Thumbnail jobs using Vidinet fail if S3 bucket requires encryption (#3354).
- Conform job against Vidinet fails if there are local-file thumbnail resources (#3374).
- Conform of video only shape fails using Vidinet (#3368).
- Conform job against Vidinet fails if defaultTimeBase not set (#3350).
- External transcoding failing when default transcoder is Vidinet (#3362).
- Document metadata extraction failing for URIs containing query parameters (#3372).
- IMF import goes into WAITING if no local transcoder exists (#3366).
- UTF-16 sidecar file fails to import from S3 (#3380).

- Export templates fail with EvaluatorException when JavaScript sandboxing is enabled (#3364).
- HTTP 500 when fetching VXA jobs using API/vxa/job (#3347).
- HTTP 404 when fetching audit logs when no username is specified (#3371).
- HTTP 400 when adding OTIF plugin on S3 (#3281).
- No username returned from job.getData("username") (#3345).
- NPE when using Aurora via CeriTalk API to analyze files (#3377).
- NPE when creating storage without specifying method URI (#3201).

#### Transcoder fixes

- MXF muxing does not fail properly fail in case of HTTP PUT errors (#3373).
- Artifacts when seeking in RED transcoded material (#3199).

#### Agent fixes

- vidispine-agent-admin command doesn't work if the agent is not binded on localhost (#2759).

#### Command line tool fixes

- KeyError from vidispine-admin if database username/password is not specified (#3209).

#### Platform

- This release adds support for running Vidispine server with PostgreSQL 9.6 (#3415).

### 18.15.6 4.14.1

*2018-08-06*

#### Improvements

- Add a "job.getUser()" function to the javascript job object (#3336).
- Return databaseSize in bytes not megabytes (#3318).
- If a file is on VSA, pick VSA transcoder even though defaultTranscoder is 'vidinet' (#3311).
- Encrypt secret keys in export locations (#3118).
- Encrypt passwords in URIs in the job metadata (#2973).
- Support using "content=external" to fetch collection external ids (#2760).

#### New agent features

- Be able to configure VSA Job Concurrency in the agent config file (#2848).

#### Agent improvements

- Be able to set VXA transcoder directAccess setting in the agent configuration file (#3332).
- Add "s3 directAccess" as the default setting in the configuration file (#3333).

### Bug fixes

- Fix possible connection issue between Vidispine cluster and VSA using websocket (#3341).
- NPE when trying to validate a vidinet shape without original tag (#3337).
- Fix thumbnail not generated during transcode jobs if thumbnail settings are specified in the shape tag (#3314).
- Fix to only use "CLOSED" or "BEING_READ" files as the source file of thumbnail jobs (#3137).

### Agent fixes

- Fix per class log level setting not working in VSA (#3340).
- VSA regression: Possible NPE when connecting to vidispine via SSH (#3334).
- Fix WebSocket file descriptor leak in VSA (#3319).
- VaaS: VSA gets into 'too many open files' issue after a while (#3312).
- Ignore broken symbolic links on VSA storages (#3176).

## 18.15.7 4.14

*2018-07-18*

### User rename

The username of users can now be *changed* (#3232). It is also possible to assign multiple usernames, or *aliases* to users. Aliases can be used to for example allow a user to log in using either the username or the users e-mail address (#3260).

```
PUT /user/stephen@example.com
Content-Type: application/xml

<UserDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <userName>stephen</userName>
  <alias>stephen@example.com</alias>
  <realName>Stephen</realName>
</UserDocument>
```

### User access keys and tokens

Support for *user access keys* has been added. Access keys are long-lived tokens that can be used to authenticate a user, that can be used instead of username/password credentials.

```
POST /user/stephen/key/
```

```
<AccessKeyDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VSID2C56CQTA7DWW7DLF</id>
  <secret>PiFHewzGN3tZ/3lSeAkBVX+5YOw3wwpHTO2iqIaa</secret>
  <status>ACTIVE</status>
  <created>2018-06-01T10:36:13.891+02:00</created>
</AccessKeyDocument>
```

To make it easy to create authentication tokens when using aliases or access keys, a new *create token resource* has been added that does not require a username.

### Group files by prefix

Files in file listings with the same prefix, in the same directory, can now be *grouped together* (#3230). This is supported for all types of storages, regardless if they have the notion of directories or not.

For example, given a file hierarchy of $YEAR/$MONTH/, we might have:

```
GET /storage/VX-1/file?prefix=true
```

```
<FileDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <file>
  ...
  </file>
  <prefixes>
    <prefix>2017/04/</prefix>
    <prefix>2017/05/</prefix>
    <prefix>2017/11/</prefix>
    <prefix>2018/02/</prefix>
  </prefixes>
</FileDocument>
```

### Checksum validation on transfer

Copy and move jobs have been updated to support `checksum validation` (#3229). If enabled, the checksum of the written file will be computed and verified against that of the source file, and if different, will cause the job to fail.

### Elasticsearch 6

Elasticsearch 6.x is now supported (#3081). Due to the removal of mapping types (https://www.elastic.co/guide/en/elasticsearch/reference/6.0/removal-of-types.html), older Elasticsearch versions (<= 5.5.x) are not supported anymore.

The support of two-way SSL (client authentication) with Elasticsearch (#3231) is added as well. It can be used in setting up PKI user authentication (https://www.elastic.co/guide/en/x-pack/current/pki-realm.html).

### Multiple audio codecs in presets

Creation of videos with audio tracks/streams with *different audio codecs* is now possible. Use the `audioTrack` element in the transcode preset (#2932). For example, to transcode audio to both AC-3 and AAC:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
    <format>mov</format>
    <audioTrack>
        <codec>ac3</codec>
        <bitrate>384000</bitrate>
    </audioTrack>
    <audioTrack>
        <codec>aac</codec>
        <bitrate>96000</bitrate>
    </audioTrack>
</TranscodePresetDocument>
```

### Thumbnail reindexing

The thumbnail index can now be rebuilt using a *re-index thumbnail request* (#3224). This indexed used to be rebuilt when rebuilding the item index, but must now be rebuild separately. This can improve the item reindex performance, especially if thumbnails are stored on an object storage such as S3 or Azure.

### AWS temporary credentials

The AWS credentials been sent to VSA can be configured using `s3CredentialType` (#2931).

### EIDR metadata import

*EIDR Integration* (#2893).

You can now import metadata from the EIDR database. There is a new resource type for accessing EIDR. Synchronization is enabled by setting the EIDR id as metadata on an item. Later on a single item or a library of items can be synchronized.

### Vidinet improvements

- Analyze shapes using Vidinet transcoder (#3219).
- Conform using Vidinet transcoder (#3220).
- Support placeholder raw import using Vidinet (#2962).

### Agent improvements

- Execute *AAF export jobs* directly on agents (#3143).

### Role changes

- Change role requirement for GET API/vxa (#3246).

### Improvements

- Add *binary* as a component type for placeholder imports (#3122).
- Support *automatic creation of users* authenticated by custom Shiro realms (#3293).
- Add a *metadatahelper* object in the common JavaScript context (#3286).
- Make it possible to configure *S3 transfers to be HTTPS* (#3227).
- Spread transfer load over available VSAs (#3126).
- Allow archive beans to clean up in case of a failed job (#3109).
- Allow archive storages to be used as destinations in storage rules (#3108).
- Add configuration to stop archived files from being used as storage rule transfer sources (#3107).
- Allow archive beans to select source storage for archive jobs (#3106).
- Include item and shape ids in job metadata for transfers started from storage rules (#3127).
- Be able to skip auto-refreshing libraries updates after item metadata changes (#3298).
- Optimize database query for metadatafield retrieval (#3295).
- Performance improvement on group deleting if a group contains many ACLs (#3270).
- Improve IMF jobs so they are compliant with Vidinet (#3258).
- Performance improvements on updating much referenced metadata (#3129).
- Use a single S3 “PUT Object” request for small files (#2956).

### New transcoder features

- Enable support for R3D files with multiple video streams (#3297).

### Transcoder improvements

- Support for Closed Captions (608 & 708) (#3052).
- Automatic creation of *thumbnails at scene changes* (#3217).

### Bug fixes

- Transcoder self test reports failed transcoders as OK (#3291).
- Transcode with destinationItem specified fails against Vidinet (#3279).
- Export jobs fail on systems with only Vidinet transcoders (#3276).
- Auto import jobs fail on systems with only Vidinet transcoders (#3274).
- Transcoder self test fails on systems with only Vidinet transcoders (#3004).
- Shape/essence import goes into WAITING if no Vidispine transcoder exists (#3269).
- Delete Jobs Finishing But Files Not Removed (#3284).
- Media check failing if apiNoauthUri contains amazonaws.com (#3255).
- Not possible to abort Atempo archive jobs (#3121).
- Item delete request with `keepShapeTagStorage` fails with NPE (#3210).
- Transcode failing when transcode preset contains multiple audio outputs specifying more channels then found in source file (#3178).
- Indexing error if multiple timespans have the same boolean field with different values (#3093).
- Scheduled requests without a body cannot be viewed (#2904).
- Parameter 'state' not optional when getting scheduled requests (#2903).
- JMS notifications fail with NameNotFoundException. (#3214).
- Item list from database becoming slower and slower (#3271).
- Thumbnail proxies are wrong when apiurl does not end with /API/ (#3262).
- Checksum computed multiple times for the same file (#3259).
- Missing subtitle text from imported SCC file (#3242).
- Incorrect output length when rendering NTSC sequence (#3240).
- Database migration fails on name column change against MySQL (#3235).
- Fix incorrect role checking on metadata dataset resources (#3198).
- Shiro plugin cannot read form parameters from servlet request (#3195).
- Fix transcode failure if one of the transcode presets has "noVideo=true" and "noAudio=true" (#2988).
- Error resuming file listing if the path contains a space (#2987).
- Agent name not set properly by vidispine-admin vxa-enable (#2399).

### Agent fixes

- VSA reports shares state incorrectly (#2980).

### Transcoder fixes

- Thumbnail job fails on file with pixelFormat yuyv422 (#3257).

### Upgrading from 4.13

- Solr: No changes to the documents. Re-indexing is not required.
- Elasticsearch: Supported versions have been changed to 5.6.x, 6.0.x, 6.1.x, or 6.2.x.
- The item re-index process will no longer also rebuild the thumbnail index that is maintained internally by VS. The thumbnail index can now instead be rebuilt separately using a *re-index thumbnail request*. To restore the old behaviour, set `disableThumbnailReindexing` to `false`.
- A number of new job steps have been added to support checksum validation on transfer.
  - COPY_FILE/MOVE_FILE, step 90 - Waiting for source file hash.
  - COPY_FILE/MOVE_FILE, step 92 - Retrieving source file hash.
  - COPY_FILE/MOVE_FILE, step 140 - Waiting for destination file hash.
  - COPY_FILE/MOVE_FILE, step 150 - Verifying file hashes.
- Previous versions had a bug where collections containing libraries did not always have recursive ACL's properly applied items in those libraries. This is fixed for newly created ACL's, but any old ones with this problem will need to be *re-indexed* with:

```
PUT /reindex/acl
```

- A role have been added for reading Vidispine Agents: `_vxa_read`. To make sure admin users receives the role properly, please run APIinit:

```
POST /APIinit
```

## 18.16 Version 4.13

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.16.1 4.13.4

*2019-03-28*

#### Bug fixes

- Move metadata migration fails if target group exists (#3708).
- Failing to reindex collection if the collection number is high (#3668).
- Incorrect normalization of combining characters for JSON (#3663).
- Infinispan cache sometimes gets cleared when a node leaves and rejoins the cluster (#3338).

#### Transcoder fixes

- Transcoder hangs on media check of tar file with encrypted MXF (#3559).
- Automatically adjust odd dimensions for H.264 output (#3414).

### 18.16.2 4.13.3

*2018-11-05*

#### Bug fixes

- Placeholder imports to the same item fail with optimistic locking errors on clustered systems (#3477).
- Importing to a full storage creates a never ending job (#3181).
- Exporting using URL with query parameters generates a metadata file without .xml extension (#3480).
- FileNotFoundException in server log when using slave licensing (#3408).
- Metadata subgroups remain in search results after item/collection is deleted (#3196).

#### Transcoder fixes

- Wrong field order information from ProRes files (#3461).

### 18.16.3 4.13.2

*2018-09-28*

#### Agent improvements

- Support checking file status on storage after transfer (#3413).
- Exclude $RECYCLE.BIN from VSA indexed directories (#3321).

#### Bug fixes

- Vidispine server opening too many UDP sockets when metrics are enabled (#3401).
- Transcoder may get more jobs than the configured "maxJobs" (#3375).
- File delete job finishing without removing file from storage (permission error) (#3284).
- NPE when creating storage without specifying method URI (#3201).

#### Transcoder fixes

- Artifacts when seeking in RED transcoded material (#3199).

### 18.16.4 4.13.1

*2018-07-06*

#### Improvements

- Be able to skip auto-refreshing libraries updates after item metadata changes (#3298).
- Optimize database query for metadatafield retrieval (#3295).
- Support automatic creation of users authenticated by custom Shiro realms (#3293).
- Performance improvement on group deleting if a group contains many ACLs (#3270).
- Improve IMF jobs so they are compliant with Vidinet (#3258).
- Performance improvements on updating much referenced metadata (#3129).
- Use a single S3 "PUT Object" request for small files (#2956).

#### New transcoder features

- Enable support for R3D files with multiple video streams (#3297).

### Bug fixes

- Item list from database becoming slower and slower (#3271).
- Thumbnail proxies are wrong when apiurl does not end with /API/ (#3262).
- Checksum computed multiple times for the same file (#3259).
- Missing subtitle text from imported SCC file (#3242).
- Incorrect output length when rendering NTSC sequence (#3240).
- Database migration fails on name column change against MySQL (#3235).
- Fix incorrect role checking on metadata dataset resources (#3198).
- Shiro plugin cannot read form parameters from servlet request (#3195).
- Fix transcode failure if one of the transcode presets has "noVideo=true" and "noAudio=true" (#2988).
- Error resuming file listing if the path contains a space (#2987).
- Agent name not set properly by vidispine-admin vxa-enable (#2399).

### Transcoder fixes

- Thumbnail job fails on file with pixelFormat yuyv422 (#3257).

## 18.16.5 4.13

*2018-04-24*

### Microsoft SQL Server 2017

Microsoft SQL Server 2017 is now supported. Make sure to use the correct *charset, collation and options* when using SQL Server (#3091).

### Codec improvements

- DNxHR encoding (#3086).
- DNxHD shape tag and license (#2886). For more information about DNxH* encoding, please see *DNxHD*.
- Update Apple ProRes libraries to latest version in order to get support for ProRes4444XQ (#3103), see *ProRes*.

### Thumbnailing using Vidinet

Vidinet transcoders can now be used to create *thumbnails*. Just specify the Vidinet resource when starting the thumbnail job to use Vidinet (#3046). Note that this only works when thumbnails are stored on S3.

```
POST /item/VX-45/thumbnail?createThumbnails=true&resourceId=VX-2
```

### Improved field validation

Metadata fields can now be restricted to a certain set of values from a specific *dataset*. Multiple fields that are restricted to the same dataset can also be validated together (#3153). Retrieve all *allowed values* for a field, or return the allowed values given that another field has a specific value:

```
GET /metadata-field/test_city/allowed-values?constraint=test_state=New%20York
```

```
<ConstraintValueListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <value id="city:ny">New York City</value>
</ConstraintValueListDocument>
```

### Command-line improvements

A retry flag has been added to the `elasticsearch init`, `db init` and `db migrate` commands. This removes the need for checking if the database/search index is online and reachable during automation (#3085).

```
$ vidispine db init --retry
$ vidispine db migrate --retry
```

The elasticsearch command has also been extended with a `check` command, similar to the `db check` command, that can be used to check if Elasticsearch needs to be initialized or not (#3016).

```
$ vidispine elasticsearch check
```

### Server configuration file improvements

The Solr endpoint can now be specified in the *server configuration file* instead of using the `solrPath` configuration property (#3072).

```
search:
  backend: solr
  url: http://solr.example.com:8983/solr/
```

### Improvements

- Add support for specifying *KMS key* (#3033).
- Key-value parameter support for *EJB notifications* (#3167).
- VSA cannot connect to VS due to too few SSH threads (#3168).
- Use database instead of search engine for batch listing (#3166).
- Display name of Vidinet service from GET API/resource/vidinet (#2989).
- Storage group query parameter to limit API results (#2952).

### Bug fixes

- Slow API/version if resources are unavailable (#2871).
- File move operation from SMB source storage doesn't delete source file (#3133).
- Invalid decoding/unescaping of smb:// credentials (#3094).
- Audio waveform start query parameter is broken (#3088).
- Storage rescan doesn't work behind a reverse proxy (#3087).
- Incorrect return code from Elasticsearch init command (#3051).
- Long restrictions on metadata fields don't work (#3083).
- Metadata locks are not locking the metadata (#3013).
- Inheritance should not be ignored on MetadataFieldGroups (#2992).
- EJB notifications don't work - beans are not found (#3090).

#### Transcoder fixes

- Fix color issue of thumbnails/posters for certain CMYK PDFs (#3188).
- Black frames bulky metadata is off-by-one after analyze (#3169).

#### Agent fixes

- Per package log setting doesn't work in VSA (#2763).

#### Platform

- This release adds support for running Vidispine server with Microsoft SQL Server 2017 (#3091).
- The PostgreSQL JDBC driver that Vidispine server is using has been updated to the latest version (#2995).

#### Upgrading from 4.12

- Solr: No changes to the documents. Re-indexing is not required.

## 18.17 Version 4.12

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.17.1 4.12.5

*2019-03-28*

#### Bug fixes

- Move metadata migration fails if target group exists (#3708).
- Incorrect normalization of combining characters for JSON (#3663).
- Infinispan cache sometimes gets cleared when a node leaves and rejoins the cluster (#3338).

#### Transcoder fixes

- Transcoder hangs on media check of tar file with encrypted MXF (#3559).
- Automatically adjust odd dimensions for H.264 output (#3414).

### 18.17.2 4.12.4

*2018-11-05*

#### Bug fixes

- FileNotFoundException in server log when using slave licensing (#3408).
- Metadata subgroups remain in search results after item/collection is deleted (#3196).

### 18.17.3 4.12.3

*2018-09-28*

### Bug fixes

- Vidispine server opening too many UDP sockets when metrics are enabled (#3401).
- Transcoder may get more jobs than the configured "maxJobs" (#3375).
- File delete job finishing without removing file from storage (permission error) (#3284).
- NPE when creating storage without specifying method URI (#3201).

## 18.17.4 4.12.2

*2018-07-06*

### Improvements

- Be able to skip auto-refreshing libraries updates after item metadata changes (#3298).
- Performance improvement on group deleting if a group contains many ACLs (#3270).
- Use a single S3 "PUT Object" request for small files (#2956).
- Optimize database query for metadatafield retrieval (#3295).
- Performance improvements on updating much referenced metadata (#3129).

### Bug fixes

- Thumbnail proxies are wrong when apiurl does not end with /API/ (#3262).
- Missing subtitle text from imported SCC file (#3242).
- Incorrect output length when rendering NTSC sequence (#3240).
- Shiro plugin cannot read form parameters from servlet request (#3195).
- Fix transcode failure if one of the transcode presets has "noVideo=true" and "noAudio=true" (#2988).
- Agent name not set properly by vidispine-admin vxa-enable (#2399).

## 18.17.5 4.12.1

*2018-04-20*

### Improvements

- Only send transcode cost estimates to Vidinet transcodes that support the source URI scheme (#2969).

### Transcoder improvements

- Enable further codec optimizations (#3079).

### Bug fixes

- Relation already exists error when running db migrate against lastest PostgreSQL (#3084).
- Better thumbnail handling for videos with varying framerate (#3076).
- Always use HTTPS for S3 pre-signed urls if S3 Server-Side Encryption is set to "aws:kms" (#3015).
- Fix APIinit schema update failure if Elasticsearch index already exists (#2996).
- Fix import cost estimate does not support all URI schemes reported by Vidinet transcoders (#2974).

### Transcoder fixes

- JPG CMYK colorshift when converting to PNG (#2979).

## 18.17.6 4.12

*2017-12-22*

### Vidinet

Transcode and shape deduction on import can now be performed using a Vidinet transcoder. This is supported for *normal imports* as well as for *auto-imports*. Image transcodes are now also supported, see the Vidinet service documentation for more detail (#2955).

### IMF support

The managing of IMF packages (IMP) has been improved.

- *Import* of assetmaps with multiple CPLs are supported, and will generate multiple shapes.
- Essence files are automatically shared between shapes, by matching UUID.
- Import of Partial IMPs are supported, and matched with existing content.
- *Export* of IMP now created up-to-date assetmaps, CPLs, PKLs.
- Upon export, subsets of virtual tracks can be selected, and *single CPLs* can be picked.

### Transcoder job limit

The number of jobs that may use a specific transcoder at the same time can now be *controlled on a per transcoder basis* (#2826). For example:

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <id>VX-7</id>
   <transcoder>
     <url>http://transcoder.example.com:8888/</url>
     <maxJob>5<maxJob>
   </transcoder>
 </ResourceDocument>
```

### Atempo Digital Archive

Files can now be archived and retrieved using *Atempo Digital Archive* (#2777).

### Metrics

Additional *metrics* exposing the online/offline status of storages, agents and resources has been added (#2806), as well as a metrics exposing the size of the cluster when running a clustered deployment (#2784).

### Shape/component selection

- It is now possible to enter shape id when importing to placeholder items with multiple placeholder shapes. This `shapeId` query parameter has been added to the *placeholder import*, *request body import*, *passkey*, and *adopt request*.
- In order to allow for concurrent uploads to components of the same type of a shape, it is now possible to specify index (1-based) of the component. The `index` query parameter has been added to the *request body import* and *adopt request*.

- When exporting item/library/collection, it is now possible to specify which tracks (components) to export, e.g., if you only want to export audio tracks, see *Exports*.
- It is now possible to *delete a certain component* from a shape.

### Component filenames

- The original filename of a component is now available in *file name scripts*.
- The component filename can also be enabled when *exporting a single shape*.

### Essence version management

- Multi-file shapes can now be *copied to newer essence versions*, for further editing.
- Essence versions can be returned based on shape tag by setting `version=latest-per-shapetag`, see *Get information*.

### Transcoder improvements

- Speed up burn-in-timecode filter (#2928).
- Prefix all temporary files with transcoder-{pid}- (#2976).
- MainConcept codec is now used to decode JPEG2000 (J2K).

### Other improvements

- Support for single part upload via request body (#2864).
- Resumable file listing for VSA and WebDAV storages (#2971).
- Improve file listing speed of WebDAV storages (#2970).
- Support for *setting headers* in the JavaScript http object (#2920).
- Allow password change using vidispine-admin from scripts (#2947).
- Support restoring S3 objects in EXPORT jobs (#2873).
- Differentiate 404 error on non-existing external id namespace (#2860).
- Be able to prevent VSA from accessing other VSA's share via vidispine server proxy (#2821).
- Add file size to checksum log message (#2789).
- Make file size available for files immediately after transfer has completed (#2787).
- Expose *job information* to filename script (#2723).

### Bug fixes

- UnsupportedOperationException when storing thumbnails on Azure (#2975).
- Vidispine cannot read blobs from Azure when blob type is "Append" (#2949).
- API/storage/{id}/rescan doesn't work on systems with port forwarding (#2944).
- APIinit overwrites customer set solrPath (#2935).
- vidispine-admin tool doesn't understand environment variables in server.yaml (#2881).
- HTTP 500 when deleting a collection or library with multiple storage rules (#2875).
- Storage methods with custom type not ignored by Vidispine (#2813).

- Unresolveable hostname in metrics configuration causes HTTP 500 from API requests (#2688).
- Import of UTF-16 encoded text file fails when file metadata parsing is enabled (#2530).
- Documentation missing for request/response representations in WADL (#2876).

#### Platform

The Apache Shiro (https://shiro.apache.org/) version *bundled with Vidispine* has been updated to version 1.4.0 (#2884).

#### Upgrading from 4.11

- Solr: No changes to the documents. Re-indexing is not required.
- All media shape deductions from jobs are now performed using asynchronous job steps. A number of new job steps have been added to support this.
  - TRANSCODE, step 400 - Finalizing media check.
  - CONFORM, step 400 - Finalizing media check.
  - AUTO_IMPORT, step 550 - Finalizing media check.
  - AUTO_IMPORT, step 1100 - Finalizing media check.
  - ESSENCE_VERSION, step 800 - Finalizing media check.
  - TIMELINE, step 400 - Creating the entities.
- To support removal of old essence files, a new job step has been added:
  - SHAPE_IMPORT, step 1000 - Remove old essence files.
- Make sure to run *APIinit* when upgrading to create the above mentioned job steps. Any existing custom job steps using these step numbers will be overwritten, so make sure to adjust the step number of any custom job step definitions using these numbers and recreate the custom steps.

## 18.18 Version 4.11

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.18.1 4.11.3

*2018-07-06*

#### Improvements

- Performance improvements on updating much referenced metadata (#3129).

#### Bug fixes

- Missing subtitle text from imported SCC file (#3242).
- Incorrect output length when rendering NTSC sequence (#3240).
- Fix transcode failure if one of the transcode presets has "noVideo=true" and "noAudio=true" (#2988).

### 18.18.2 4.11.2

*2018-04-20*

#### Transcoder improvements

- Enable further codec optimizations (#3079).

#### Bug fixes

- Relation already exists error when running db migrate against lastest PostgreSQL (#3084).
- Better thumbnail handling for videos with varying framerate (#3076).
- Fix NPE when talking to the CeriTalk API in Aurora (#3040).

#### Transcoder fixes

- JPG CMYK colorshift when converting to PNG (#2979).

### 18.18.3 4.11.1

*2017-12-22*

#### Transcoder improvements

- Improved handling of MXF files with Open Partitions (#2951).
- Enable higher AAC bitrates (#2929).
- Support image overlays also for conform jobs (#2910).
- Improved memory handling in transcoder (less heap usage) (#2872).

#### Bug fixes

- Silent audio channels when rendering item with discrete audio components (#2967).
- Demuxer settings from preset not used for conform jobs (#2934).
- Frame rate and audio layout different across conform and transcode jobs (#2923).
- Reindexing collections stops certain item searches from working (#2917).
- StackOverflowError when changing access on collection with many grandchild-collections (#2909).
- Burn-in timecodes not working with conform jobs (#2883).

#### Transcoder fixes

- DNx Transcodes not working (#2937).
- Transcode progress missing from conform jobs (#2905).
- Bug in H.264 generation at high bitrates (#2894).
- Files encoded with Vidispine cause a crash in Avid MediaComposer (#2888).
- Transcoder media time is incorrect for AES3 audio (#2885).
- Incorrect resolution of Canon Raw file (CR2) (#2877).

### 18.18.4 4.11

*2017-10-13*

### Vidinet

This release brings support for using services from our new media services platform Vidinet (https://www.vidispine.com/vidinet). Both *transcode* and *quality control* services are supported, and you can of course request *cost estimates* before using the services (#2781).

```
GET /resource/vidinet
```

```
<ResourceListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <resource>
    <id>VX-1024</id>
    <vidinet>
      <url>vidinet://key:secret@7a104fb2-e520-430d-b6e2-b8acc9cbc588</url>
      <endpoint>https://services.vidinet.net</endpoint>
      <type>TRANSCODER</type>
      <state>ONLINE</state>
    </vidinet>
  </resource>
</ResourceListDocument>
```

### Private key authentication for SFTP

Authentication using private keys is now supported for SFTP. This is supported for both storages and transfers (#2799).

### Documentation in WADL

The application WADL has been updated to include documentation for methods and query and matrix parameters (#2832).

### S3 part size selection

The S3 upload `part size` is now automatically increased to better handle uploads of large files. Files up to 3.5 TB are now supported without any further configuration (#2846).

### Improvements

- Include external ids in *transient metadata* (#2855).
- Add support for selecting transcoder resource on *transcode* (#2801).
- Add support for selecting transcoder resource on *shape analysis* (#2828).
- Enable multiple VSAs computing hashes `concurrently` (#2778).
- Be able to *modify a notification* (#2193).

### Bug fixes

- Media check of file on SFTP storage never finishing (#2822).
- Cannot export file to directory that does not exist using SFTP (#2823).
- Missing role checks for item resource (#2815).
- Slow library item metadata update jobs (#2603).
- Add hashing algorithm in the file hash notification (#2841).
- SCC and srt sidecar files should not be auto-imported by default (#2833).
- Indexing halting when adding a date field with index=extend (#2788).

- Not able to update group by UUID if there are multiple groups with the same name (#2761).
- Search for groups not containing field return internal server error (#2652).
- NPE on import if all storages are full (#2861).
- Empty timestamps returned when using path projections (#2858).

#### Transcoder fixes

- Transcode failing with numberOfPackets error on auto-import if fast start is enabled (#2798).

#### Upgrading from 4.10

- Solr: No changes to the documents. Re-indexing is not required.

## 18.19 Version 4.10

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.19.1 4.10.4

*2018-07-06*

#### Improvements

- Performance improvements on updating much referenced metadata (#3129).

#### Bug fixes

- Missing subtitle text from imported SCC file (#3242).
- Incorrect output length when rendering NTSC sequence (#3240).
- Fix transcode failure if one of the transcode presets has "noVideo=true" and "noAudio=true" (#2988).

### 18.19.2 4.10.3

*2018-04-20*

#### Transcoder improvements

- Enable further codec optimizations (#3079).

#### Bug fixes

- Relation already exists error when running db migrate against lastest PostgreSQL (#3084).
- Fix NPE when talking to the CeriTalk API in Aurora (#3040).

#### Transcoder fixes

- JPG CMYK colorshift when converting to PNG (#2979).

### 18.19.3 4.10.2

*2017-12-22*

### Transcoder improvements

- Improved handling of MXF files with Open Partitions (#2951).
- Enable higher AAC bitrates (#2929).
- Support image overlays also for conform jobs (#2910).
- Improved memory handling in transcoder (less heap usage) (#2872).

### Bug fixes

- Silent audio channels when rendering item with discrete audio components (#2967).
- Demuxer settings from preset not used for conform jobs (#2934).
- Frame rate and audio layout different across conform and transcode jobs (#2923).
- Reindexing collections stops certain item searches from working (#2917).
- StackOverflowError when changing access on collection with many grandchild-collections (#2909).
- Burn-in timecodes not working with conform jobs (#2883).

### Transcoder fixes

- Transcode progress missing from conform jobs (#2905).
- Bug in H.264 generation at high bitrates (#2894).
- Incorrect resolution of Canon Raw file (CR2) (#2877).

## 18.19.4 4.10.1

*2017-10-06*

### Improvements

- Save text and outline color to item metadata on STL sidecar import (#2785).

### Transcoder improvements

- Support new H.264 FourCCs (#2819).

### Bug fixes

- Infinispan stale locks problem (#2775).
- Possible Solr query errors on autocomplete on string-exact fields (#2865).
- Imports using AWS presigned URLs fail with HTTP 403 (#2863).
- Incorrect container component after auto-import of growing file (#2859).
- Failed to delete an item if its bulky metadata contain null values (#2857).
- Raw import with “ids” parameter fails to set external id (#2847).
- Incorrect data in file, storage and metadata notification (#2838).
- Possible failing conform job if the original file is lost (#2831).
- Incorrect result on wildcard search for string-exact if the value contains a space (#2796).
- HTTP 500 when searching for string-exact field with text containing AND/OR (#2793).

- Auto-completion not case insensitive for string-exact fields (#2780).

### Transcoder fixes

- Avoid overflow in aspect ratio calculation (#2869).
- NPE (SIGSEGV) in Transcoder when reading MXF files (#2868).
- Start timecode is not set with correct in NLEJob (#2867).
- Signed integer problem with startTimecode (#2862).
- Exception occured while opening wmv muxer for.... (#2850).

### Agent fixes

- File transfer job in agent not using settings from agent conf (#2866).
- VSA fails to return transcoder status for job that is finishing (#2807).

## 18.19.5 4.10

*2017-07-28*

### Elasticsearch

- *Elasticsearch support* (#2628). Elasticsearch can be used instead of Solr for *searching*. Certain *restrictions* apply.

### Image sequence generation

- Image sequence output (#2726). The transcoder generates image sequences for shape tag with transcoder preset codec `tiff`, `png`, or `dpx`, e.g.:

```
<TranscodePresetDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <format>tiff</format>
  <audio>
  <noAudio>true</noAudio>
  </audio>
  <video>
    <codec>tiff</codec>
  </video>
  <sequenceOutput/>
</TranscodePresetDocument>
```

### Universal storage method

Vidispine is able to manage files that are not under a common root using a *universal storage*.

And it's also possible to import files to a universal storage *without firstly copy it to another storage* (#2719).

### Improvements

- Allow owner of an item/collection to be changed (#2706).
- Support removal of values by actual value (#2635).
- Add option when updating metadata group so that resulting group exactly matches input XML (#2747).
- Configurable transcoder DNS cache TTLs (#2712).

- Filter jobs by completion date (#2702).
- S3 lifecycled bucket as ARCHIVE storage (#2697).
- Expose the source storage for COPY_FILE jobs (sourceStorageId) (#2691).

### Transcoder improvements

- Faster MXF parsing in transcoder (#2753).

### Bug fixes

- Slow disk reads causing optimistic locking errors on import (#2756).
- JDBC connection not returned to pool on thread interrupt/XA exception (#2754).
- Storage method with type "AUTO" not selected when requesting an auto method (#2750).
- Excessive logging for empty storages (#2730).
- Excessive DNS logging when DNS entry missing (#2727).
- NPE from auto-import job after deleting source storage (#2718).
- Transcoder transfer only performed for local file transfers (#2715).
- File size missing in job progress for copy jobs performed by agent (#2714).
- Cannot copy files between S3 buckets in different regions (#2703).
- GET /job/(id) accept:text/plain gives 500 Internal Server Error (#2700).
- GET/DELETE /resource/{resourcetype}/{id} should validate that id is of type resourcetype (#2696).
- Aborted jobs not purged from the database (#2692).
- POST /storage/abc/rescan gives Internal Server Error (#2681).
- No notification sent for file delete on readonly storage (#2675).
- Create storage where method metadata key is missing gives 503 Service Unavailable (#2655).
- POST /item/{id1}/relation/{id2} without direction gives NullPointerException (#2643).

### Transcoder fixes

- Transcoder dies when creating overlay on video (#2667).
- Job fails on certain sequence renderings (#2779).

### Agent fixes

- Agent proxy URIs lost when jobs are accepted at the same time (#2717).
- No progress for transfer jobs on agent (#2713).
- Agent does not support named transcoder licenses (#2711).

### Upgrading from 4.9

- Solr: No changes to the documents. Re-indexing is not required.

## 18.20 Version 4.9

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.20.1 4.9.3

*2017-12-22*

#### Bug fixes

- Silent audio channels when rendering item with discrete audio components (#2967).
- Frame rate and audio layout different across conform and transcode jobs (#2923).

#### Transcoder fixes

- Incorrect resolution of Canon Raw file (CR2) (#2877).

### 18.20.2 4.9.2

*2017-10-06*

#### Bug fixes

- JDBC connection not returned to pool on thread interrupt/XA exception (#2754).
- Infinispan stale locks problem (#2775).
- Possible Solr query errors on autocomplete on string-exact fields (#2865).
- Imports using AWS presigned URLs fail with HTTP 403 (#2863).
- Incorrect container component after auto-import of growing file (#2859).
- Failed to delete an item if its bulky metadata contain null values (#2857).
- Incorrect data in file, storage and metadata notification (#2838).
- Incorrect result on wildcard search for string-exact if the value contains a space (#2796).
- HTTP 500 when searching for string-exact field with text containing AND/OR (#2793).
- Auto-completion not case insensitive for string-exact fields (#2780).

#### Transcoder fixes

- Signed integer problem with startTimecode (#2862).

### 18.20.3 4.9.1

*2017-07-12*

#### Improvements

- Allow eu-west-2 to be specified as region for S3 (#2772).

### Bug fixes

- Content-Range header error when uploading file larger than 2GB to Google Cloud Storage (#2709).
- Waveform not created for audio-only file (#2751).
- License check failing with SocketException (ioctl SIOCGLIFFLAGS failed) (#2770).
- Packing list not properly detected from IMF (#2771).
- Sequence render job failing due to incorrect media type error (#2774).
- Cannot create storage with multiple image sequence definitions (#2765).
- Center aligned subtitles come out as left aligned after STL sidecar import (#2693).
- Incorrect mime type of .ai (Adobe Illustrator) file (#2588).

### Transcoder fixes

- TTML import via MXF broken (#2769).
- Corrupt mp4 output created (#2768).
- Time code does not reflect NTSC frame rate (#2767).
- Interlacing setting is not respected in NLEJob (#2766).

### Agent fixes

- Conform job URLs not proxied by agent (#2773).
- Overlay address not proxied by agent (#2687).

## 18.20.4 4.9

*2017-04-03*

### Service configuration

It is now possible to configure if *Vidispine services* should be *allowed to run* on a specific server instance (#2671). For example, in a clustered setup, some instances could be configured to only serve API requests by disabling all background services:

```
services:
  disabled: all
```

### Glacier improvements

Files on S3 buckets that have been archived on Glacier using lifecycle rules will now automatically be restored when copied between storages (#2597).

It is now also possible to configure the S3 retrieval policy for faster restores from Glacier (#2586). This can be configured *per job* or *per storage* using the `retrievalTier` parameter.

### OpenAPI 2 compatibility

The API can now be configured to accept query parameters where matrix parameters are expected (#2684). This can make it easier to use Vidispine with clients/specifications that do not support matrix parameters, such as OpenAPI 2.

```
api:
  openApi2Compatible: true
```

### Improvements

- Allow *deletion of individual jobs* (#2579).
- Allow *auto-import rules to be disabled* (#2553).
- Ability to *exclude self-tests* by name (#2540).
- Allow multiple URIs to be specified in an export location (#2679).
- Be able to *delete job metadata by key* from JavaScript (#2609).
- Support partial file restore with DIVA (#2537).

### Transcoder improvements

- Increase read buffer length for improved S3 performance (#2683).
- Quicker shape deduction for streams without timecode (#2682).
- Support all Avid DNxHD and DNxHR profiles when decoding (#2686).
- Handling of different KAG values in MXF (#2685).

### Agent improvements

- Make log rotation optional in agent (#2626).
- Add BlackPearl "physical placement" info to file metadata (#2607).

### Bug fixes

- Agent registration failing when schema validation is enabled (#2678).
- Agent registration failing on system with DNS transcoder directory (#2665).
- Transcoder metrics not exposed for DNS discovered transcoders (#2657).
- Transcode in raw import waiting 5 seconds for preceding steps to finish (#2656).
- Cannot use segment files with URI containing query parameters (#2651).
- Transcoder directAccess not applied to DNS discovered transcoders (#2647).
- Move to BlackPearl deletes source file too quickly (#2615).
- Self tests fail with HTTP 500 when schema validation is enabled (#2612).
- NPE when indexing item with empty thumbnail database on S3 (#2605).
- Query parameter "id" in GET storage/file does not work with multiple ids (#2583).
- Missing role check for PUT API/storage/file/{id}/abandon (#2556).
- Rollback not performed if validation fails when creating a storage (#2550).
- Authentication token not invalidated after user password change (#2542).
- Authentication token not invalidated when user is disabled (#2541).
- Glacier archive job fails with "Invalid Content-Range" error (#2504).
- Reject S3 buckets with names that are not valid hostnames (#2416).

### Transcoder fixes

- Transcode of PSD failing with CompressionNotSupported (#2601).

#### Agent fixes

- Agent fails to start if default port is in use (#2680).
- All jobs returned from agent when fetching running transcoder jobs only (#2658).
- Slow media checks when using agent as a transcoder proxy (#2654).
- Agent not deleting temp files when using segment files (#2653).
- Incorrect proxying of URIs containing query parameters (#2650).

#### Removed in 4.9

Support for Ubuntu 12.04 has been discontinued.

#### Upgrading from 4.8

- Solr: No changes to the documents. Re-indexing is not required.
- Support for Ubuntu 12.04 has been discontinued.

## 18.21 Version 4.8

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.21.1 4.8.3

*2017-10-06*

#### Bug fixes

- JDBC connection not returned to pool on thread interrupt/XA exception (#2754).
- Infinispan stale locks problem (#2775).
- Imports using AWS presigned URLs fail with HTTP 403 (#2863).
- Incorrect container component after auto-import of growing file (#2859).
- Failed to delete an item if its bulky metadata contain null values (#2857).
- Incorrect data in file, storage and metadata notification (#2838).
- Incorrect result on wildcard search for string-exact if the value contains a space (#2796).
- HTTP 500 when searching for string-exact field with text containing AND/OR (#2793).
- Auto-completion not case insensitive for string-exact fields (#2780).

### 18.21.2 4.8.2

*2017-07-12*

#### Bug fixes

- License check failing with SocketException (ioctl SIOCGLIFFLAGS failed) (#2770).
- Center aligned subtitles come out as left aligned after STL sidecar import (#2693).
- Incorrect mime type of .ai (Adobe Illustrator) file (#2588).

### Transcoder fixes

- Corrupt mp4 output created (#2768).
- Time code does not reflect NTSC frame rate (#2767).
- Interlacing setting is not respected in NLEJob (#2766).

### Agent fixes

- Overlay address not proxied by agent (#2687).
- All jobs returned from agent when fetching running transcoder jobs only (#2658).

## 18.21.3 4.8.1

*2017-03-31*

### Bug fixes

- Thumbnails not created for .pcd files (#2617).
- Thumbnails not working for .eps files (#2596).
- Transfers from vxa:// uri to file:// does not work (NPE) (#2616).
- Object on tape-only not restored properly from BlackPearl 3.x (#2670).
- PUT or POST from JavaScript fails if no input is specified (#2660).
- Incorrect noauth-URL returned from API/poster (#2666).

### Transcoder fixes

- Transcode using partial file never finishes (#2571).
- Transcoder crashes when using burnTimecode=true (#2642).
- Transcoder crash on converting MP3 to JPEG (!) (#2625).
- Transcoder crash on JPEG input (#2624).
- Incorrect H264 level when using CBR (#2674).
- Binary component not created for .srt files (#2661).
- Transcoder does not recognize <setting><key>level</key> (#2646).
- Duration analysis failing on audio-only MOV files (#2611).
- Transcoder fails to calculate PTS to DTS ratio (I/O error2 - 22) (#2636).
- Transcoder accessing files in parent folder (#2608).

### Agent fixes

- Agent not finding files with brackets, %20 and %21 in file name (#2610).

## 18.21.4 4.8

*2017-01-12*

### 8K RED support

The transcoder has been updated to support 8K RED files (#2581).

### Amazon S3 Transfer Acceleration

Improve transfer speeds using Amazon S3 Transfer Acceleration (http://docs.aws.amazon.com/AmazonS3/latest/dev/transfer-acceleration.html) (#2308). Enable using the *acceleration parameter*.

### Aspera FASP storage transfers

Transfers between storages can now be made using *Aspera FASP* (#1508).

### Improvements

- Be able to specify the target storage for essence imports (#2529).
- Delete multiple *items* and *collections* in a single request (#2517).
- Allow *duplicate item relations* to be rejected (#2516).
- Allow *duplicate access control entries* to be rejected (#2227).
- Support *exclusive range queries* (#2515).
- Support *document metadata notifications* (#2493).
- Be able to `skip sidecar import` during auto import (#2444).
- Do not re-index an item if adding/removing it to a collection is a no-op (#2430).
- Set collection contents and metadata *on create* (#2297).
- Configurable Glacier archive description (#2063). See `glacierArchiveDescription`.

### Bug fixes

- Update BlackPearl SDK to 3.2.8 (#2578).
- Incorrect job problem for import to specific storage (#2555).
- GET API/document does not show first document (#2554).
- Thumbnail jobs failing for image sequences (#2547).
- Transaction timeout when letting agent compute checksums (#2546).
- Hits not shown for API/document (#2535).
- Using “p=metadata.timespan.group.uuid” when searching for items can cause an NPE (#2524).
- Cannot fail waiting job from JavaScript (#2456).
- Cannot get item field-group using external id (#2455).
- Missing role check for DELETE API/storage/file/{id}/entity (#2454).
- Incorrect media type for image sequences (#2428).
- Shape import creates file on wrong storage (#2231).
- Cannot start placeholder import using external id (#2226).
- VS creating copy jobs to read-only storages (#2225).

### Transcoder fixes

### Agent fixes

- Reconnecting VSA to SSH-less VS may lead to NPE in VS (#2590).

#### Platform

- This release adds support for running the server agent on Ubuntu 16.04 (#2568).

#### Upgrading from 4.7

- Solr: No changes to the documents. Re-indexing is not required.

## 18.22 Version 4.7

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.22.1 4.7.4

*2017-10-06*

#### Bug fixes

- JDBC connection not returned to pool on thread interrupt/XA exception (#2754).
- Infinispan stale locks problem (#2775).
- Imports using AWS presigned URLs fail with HTTP 403 (#2863).
- Incorrect container component after auto-import of growing file (#2859).
- Failed to delete an item if its bulky metadata contain null values (#2857).
- Incorrect data in file, storage and metadata notification (#2838).
- Auto-completion not case insensitive for string-exact fields (#2780).

### 18.22.2 4.7.3

*2017-07-12*

#### Bug fixes

- Center aligned subtitles come out as left aligned after STL sidecar import (#2693).
- Incorrect mime type of .ai (Adobe Illustrator) file (#2588).

#### Transcoder fixes

- Corrupt mp4 output created (#2768).
- Time code does not reflect NTSC frame rate (#2767).
- Interlacing setting is not respected in NLEJob (#2766).

#### Agent fixes

- Overlay address not proxied by agent (#2687).
- All jobs returned from agent when fetching running transcoder jobs only (#2658).

### 18.22.3 4.7.2

*2017-03-31*

### Bug fixes

- Transfers from vxa:// uri to file:// does not work (NPE) (#2616).
- Object on tape-only not restored properly from BlackPearl 3.x (#2670).
- PUT or POST from JavaScript fails if no input is specified (#2660).
- Incorrect noauth-URL returned from API/poster (#2666).

### Transcoder fixes

- Transcoder crashes when using burnTimecode=true (#2642).
- Transcoder crash on converting MP3 to JPEG (!) (#2625).
- Transcoder crash on JPEG input (#2624).
- Transcoder stuck on media check of ISO files (high swapfile usage) (#2505).
- Incorrect H264 level when using CBR (#2674).
- Binary component not created for .srt files (#2661).
- Transcoder does not recognize <setting><key>level</key> (#2646).
- Duration analysis failing on audio-only MOV files (#2611).
- Transcoder fails to calculate PTS to DTS ratio (I/O error2 - 22) (#2636).
- Transcoder accessing files in parent folder (#2608).

### Agent fixes

- Agent not finding files with brackets, %20 and %21 in file name (#2610).

## 18.22.4 4.7.1

*2016-12-21*

### Transcoder improvements

- Add option to not remove padding from IMX packets in MOV (#2591).

### Agent improvements

- Have option not to create storages in Vidispine (#2594).
- Do not use the "instance host" field for shares with VSA (#2593).
- FTP connections should be reused (#2575).

### Bug fixes

- Update BlackPearl SDK to 3.2.8 (#2578).
- Slow SFTP transfers (#2564).
- Incorrect media type for ProTools (.ptx) files (#2558).
- Server logs do not show meaningful error when VSA fails to compute checksum (#2557).
- NPE when setting indextimespans=false and using item field groups (#2545).
- BlackPearl writes failing if object prefix does not exist (#2531).

- File permission self test failing on CentOS 6 (#2565).
- File ETag not in double quotes (#2513).
- Excessive logging on slave license validation error (#2484).

#### Transcoder fixes

- High memory usage of idle transcoder (mmap threshold) (#2569).
- Memory leak with multi-page images (#2573).
- Huge transcoder log files (#2570).

#### Agent fixes

- Reconnecting VSA to SSH-less VS may lead to NPE in VS (#2590).
- ValueError when adding S3/DS3 network share on CentOS 6 (#2567).
- VSA remove-share not removed from Vidispine Server (#2589).
- Shares cannot be removed in VSA mode (#2563).
- Agent disconnecting due to buffer underflow (#2562).
- Requests to agent fail with HTTP 415 Unsupported media type (#2560).
- Username and password shown in VXA log (#2534).

### 18.22.5 4.7

*2016-10-19*

#### External storage of private keys and secrets

Storage credentials and private keys can now be stored in an *external location* and no longer need to be embedded in the storage URI (#2482).

#### Google Cloud Storage

Google Cloud Storage buckets can now be used as storages and with any job accepting a URI. See the *gs scheme* for details (#2322).

#### MatrixStore 3.2

The MatrixStore SDK has been updated from version 3.1 to version 3.2. This allows Vidispine to connect to MatrixStore 3.2 clusters. It is still possible to use the 3.1 SDK instead by installing an optional subpackage, see *Upgrade notes* (#2437).

#### Command line tools

The *vsctl* command line tool can be used to query, inspect and update Vidispine (#2436). For example, to get a glance of the status of a system:

```
$ vsctl -h vs.example.com status
http://vs.example.com:8080
Version: 4.7
Site: VX
License: valid
```

```
 1 online transcoders
 12 online storages, 5 offline
 0 pending jobs, 7 waiting jobs, 0 job problems

Status:

- Capacity of storage VX-2671 is above the high watermark and needs cleanup
```

### HTTP/2

Dropwizard has been updated to 1.0 which uses Jetty 9.3 (#2432). This adds support for the HTTP/2 over TLS (http://www.dropwizard.io/1.0.2/docs/manual/configuration.html#http-2-over-tls) (`h2`) and HTTP/2 Plain Text (http://www.dropwizard.io/1.0.2/docs/manual/configuration.html#http-2-plain-text) (`h2c`) server connectors:

```
- type: h2c
  port: 8088
- type: h2
  port: 8081
  keyStorePath: /etc/vidispine/server.keystore
  keyStorePassword: example
  validateCerts: false
  validatePeers: false
```

### Improvements

- IMF *read and proxy generation* (#2319).
- Notification publishing to *Amazon SQS* (#2316).
- Support *64-bit integers* in metadata (#2345).
- *Update collection content* in a single request (#2392).
- Key-value metadata for *collection-item relationships* (#1173).
- Support for *text overlays* in video and images (#2488).
- *Cancel/pause reindexing* using the API (#1815).
- Select subtitle language when *rendering a sequence* (#2485).
- Make it possible to *create a file and write data* to it in one API call (#2413).
- Add endpoint to *delete all shapes* for an item (#2393).
- Support HEAD for noauth streaming URLs (#2495).
- Expose embedded broker *queue sizes as metrics* (#2490).
- Support growing files in VSA (#2441).
- Support fetching files with a *specific state* (#2438).
- Allow target storage to be specified for *shape imports* (#2417).
- Cacheable *saved search* endpoint (#2412).
- Support for *.scc subtitle import* and *export* (#2294).

#### Transcoder improvements

- Support for transcoding image sequences in ARRIRAW or OpenEXR format (#2499). Note that performance is quite slow.
- Support for transcoding image sequences in DNG format. Note that performance is quite slow (#2429).
- Improved rendering of Arabic subtitles (#2427).

#### Bug fixes

- Thumbnails not deleted from S3 storage (#2523).
- Library updates slowing down indexing (#2494).
- Incorrect result when listing files with "&" in path, using the "path" parameter (#2483).
- Schema validation not performed for item with assigned field group (#2423).
- Possible duplicate document entries due to concurrent creations (#2414).
- Transcode job retry failing with missing file error (#2400).
- Path projection not working for inherited metadata (#2419).
- PUT API/item/(id)/sequence/vidispine does not accept JSON (#2397).
- Exception stack traces in server log if license server is offline (#2500).
- Self test failing on transcoder directory resource (#2489).
- Transcode self test not using configured reverse address (#2487).
- Thumbnail self test fails for direct+ URLs (#1642).
- No websocket updates for new VXA registrations (#2447).

#### Transcoder fixes

- Transcoder updates modification time on input file (#2460).

#### Agent fixes

- When VSA is reconnecting, running transcode jobs cannot connect to same upload port (#2522).

#### Upgrading from 4.6

- Solr: The Solr schema must be updated to use the new *long integer datatype*.
- Solr: No changes to the documents. Re-indexing is not required.
- Version 3.2 of the MatrixStore SDK is now installed by default. The *vidispine-server-matrixstore3.1* package must be installed to connect to MatrixStore 3.1.

## 18.23 Version 4.6

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.23.1 4.6.5

*2017-07-12*

### Bug fixes

- Center aligned subtitles come out as left aligned after STL sidecar import (#2693).
- Incorrect mime type of .ai (Adobe Illustrator) file (#2588).

### Transcoder fixes

- Time code does not reflect NTSC frame rate (#2767).
- Interlacing setting is not respected in NLEJob (#2766).

### Agent fixes

- Overlay address not proxied by agent (#2687).
- All jobs returned from agent when fetching running transcoder jobs only (#2658).

## 18.23.2 4.6.4

*2017-03-31*

### Bug fixes

- Object on tape-only not restored properly from BlackPearl 3.x (#2670).
- PUT or POST from JavaScript fails if no input is specified (#2660).
- Incorrect noauth-URL returned from API/poster (#2666).

### Transcoder fixes

- Incorrect H264 level when using CBR (#2674).
- Binary component not created for .srt files (#2661).
- Transcoder does not recognize <setting><key>level</key> (#2646).
- Duration analysis failing on audio-only MOV files (#2611).
- Transcoder fails to calculate PTS to DTS ratio (I/O error2 - 22) (#2636).
- Transcoder accessing files in parent folder (#2608).

### Agent fixes

- Username and password shown in agent log file (#2534).

## 18.23.3 4.6.3

*2016-12-21*

### Transcoder improvements

- Add option to not remove padding from IMX packets in MOV (#2591).

### Bug fixes

- Update BlackPearl SDK to 3.2.8 (#2578).
- Slow SFTP transfers (#2564).
- Incorrect media type for ProTools (.ptx) files (#2558).
- Server logs do not show meaningful error when VSA fails to compute checksum (#2557).
- NPE when setting indextimespans=false and using item field groups (#2545).
- BlackPearl writes failing if object prefix does not exist (#2531).

### Transcoder fixes

- High memory usage of idle transcoder (mmap threshold) (#2569).
- Memory leak with multi-page images (#2573).
- Huge transcoder log files (#2570).
- Transcoder updates modification time on input file (#2460).

### Agent fixes

- Reconnecting VSA to SSH-less VS may lead to NPE in VS (#2590).
- ValueError when adding S3/DS3 network share on CentOS 6 (#2567).

## 18.23.4 4.6.2

*2016-10-13*

### Bug fixes

- Export job does not properly save files from being deleted (#2510).
- Slow reindex of standalone files (#2507).
- Library endpoints do not work with long identifiers (#2492).
- Export job creates sidecar XML with extension .xml.xml when using filename script (#2473).
- Cannot copy file between S3 and local S3 compatible storage (#2389).
- NPE on container raw passkey import (#2433).
- Waiting raw passkey import job not aborting (#2342).

### Packaging fixes

- Config file /etc/sysconfig/vidispine and license replaced on upgrade (#2497).

### Transcoder fixes

- Black video in Safari (#2511).
- D-10 VITC lines should be removed from poster (#2509).
- Transcoder does not handle content without aspect ration information (#2508).
- Don’t create thumbnails for all layers in an image (#2472).

### Agent fixes

- VSA is sometimes creating SSH connections that are not released (#2514).
- ConcurrentModificationException on agent reconnect (#2448).

## 18.23.5 4.6.1

*2016-09-01*

### Improvements

- Delete an *external id for an entity* (#1138).
- Metadata performance improvements (#2424).

### Transcoder improvements

- Improved handling of MXF files with incorrect footer position offsets (#2468).

### Bug fixes

- Index checker holding lock on sequence table (#2370).
- Transient metadata of collection not updated when item is deleted (#2373).
- Incorrect query for phrase search containing quotation marks (#2375).
- "noescape" is affecting the result of version 2 search documents (#2376).
- Invalid range request exceptions with DS3 (#2406).
- Task servlets stops running without any error (#2410).
- Possible NPE when reindexing items with thumbnails on S3 (#2418).
- Transcode of a image sequence failed if the sequence number is padded (#2420).
- VSA import fails if the file path contains a space (#2421).
- VSA import fails if Vidispine server is restarted (#2422).
- Image sequence export jobs failing with FAILED_TOTAL (#2426).
- /APInoauth/stitch error - Comparison method violates its general contract! (#2431).
- Regression: "s3ProxyValidTime" setting not working (#2434).
- Deletion of entities should delete associated external IDs (#2443).
- Possible NPE on EVS sidecar import (#2445).
- Race between copy job and storage worker causing job to fail (#2462).
- Demuxer settings not passed to transcoder for thumbnail jobs (#2461).

### Transcoder fixes

- Transcode failing for MPEG2 with truncated time base value (#2465).
- Transcoder crashing on transcode of DNxHD with mixed interlaced and progressive frames (#2466).
- Incorrect frame rate in WMV header for 30 FPS material (#2467).
- Out-of-sync output for 23.967 MPEG PS (#2469).
- Interlaced H264 incorrectly reported as progressive (#2470).

- Black video in Safari for transcoded 29.97 FPS H264 (#2471).
- Broken encoding of AAC in MP4 (#2425).
- D10 blanking lines included in posters (#2463).
- Incorrect scaling of posters from NTSC files (#2464).

## 18.23.6 4.6

*2016-06-30*

### Read-only instances

An instance can now be connected to a read-only database such as an Amazon RDS Read Replica (https://aws.amazon.com/rds/details/read-replicas/sph), or a MySQL or PostgreSQL replica (#2292).

```
database:
  type: read_only_replica
```

This can also be used to connect to a database snapshot that should not be modified.

```
database:
  type: read_only_snapshot
```

Request- and search logging is still enabled by default when connected to a read replica/snapshot. This works by having updates submitted to a read-write instance using ActiveMQ. To disable this, use:

```
api:
  requestLogging: false
  searchHistory: false
```

### Image sequences

Image sequences can now be imported using the normal import jobs, or directly from a storage. When importing using a URI, the URI must contain a file fragment and a wildcard specifying the placement of the sequence number (#2296). See *Image sequences*.

```
POST API/import?uri=file:///srv/media/take1/*.png#file=00000-15000
```

### CloudConvert

CloudConvert (https://cloudconvert.com/) can be used by Vidispine as an alternative transcoder. This is supported in all import and transcode jobs. See *CloudConvert*.

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <cloudconvert>
    <apiKey>your-api-key</apiKey>
  </cloudconvert>
</ResourceDocument>
```

```
POST /item/(item-id)/transcode?jobmetadata=useCloudConvert=true
```

### Storage load balancing

Files can now be *load balanced* across multiple selected storages (#2149). For example, to have a file stored on at least two of the storages from a specific group:

```
<StorageRuleDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <storageCount>2</storageCount>
  <pool>
    <group>local-storage-pool</group>
  </pool>
</StorageRuleDocument>
```

Additional storages could then be added to this group in case additional storage space is required.

### Search improvements

- Configurable *facet limit and counts* (#2341).
- Support *autocomplete while searching* (#2351).

### VSA in cluster configuration

- Support *VSA* connecting to Vidispine in a cluster configuration.

### Other improvements

- Define your own *custom job types* (#2379).
- Threaded notification deliveries (#2192).
- *Read-only/offline thumbnail resources* (#2327).
- Make it possible to *delete a user* completely (#2381).
- Add document and PDF *media types* (#2357).
- Make it possible to set username in auto import rule (#2356).
- Quota rule *filtering* (#2353).
- Support for querying collection *child item count* (#2350).
- Add configuration property to control *agent sync behaviour* (#2382).
- Support useOriginalFilename=true with copy/move jobs (#2344).
- Setting *Content-Disposition headers* for S3 when signed URLs are generated (#2326).

### Bug fixes

- Solr solrUpdateQueueSize setting not working (#2396).
- XMP sidecar import not working if parseXMP=false (#2354).
- MxfServerAccessControlBean uses a fixed connection ID (#2378).
- Add multipart/form-data noauth import resource (#2380).

### Platform

- This release adds support for Ubuntu 16.04 and MySQL 5.7 (#2321).
- The transcoder no longer auto-starts when installed, and will restart if already during upgrade (from 4.6 and forward).

#### Deprecated in 4.6

- Image sequence auto-import using *auto-import rule settings*.
- Image sequence *placeholder import* using `type=image-sequence` or `type=dpx`.

#### Discontinued in 4.6

- Support for GlassFish has been discontinued. Use *Installation* instead.
- Support for Java 7 has been discontinued. Use Java 8 instead.
- Support for Microsoft Windows Server has been discontinued.

#### Upgrading from 4.5

- Solr: No changes to the documents. Re-indexing is not required.
- Support for GlassFish has been discontinued. Use *Installation* instead.
- Support for Java 7 has been discontinued. Use Java 8 instead.
- Support for Microsoft Windows Server has been discontinued.

## 18.24 Version 4.5

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.24.1 4.5.9

*2017-07-12*

#### Bug fixes

- Center aligned subtitles come out as left aligned after STL sidecar import (#2693).
- Incorrect mime type of .ai (Adobe Illustrator) file (#2588).

#### Agent fixes

- Overlay address not proxied by agent (#2687).
- All jobs returned from agent when fetching running transcoder jobs only (#2658).

### 18.24.2 4.5.8

*2017-03-31*

#### Bug fixes

- Incorrect noauth-URL returned from API/poster (#2666).

#### Transcoder fixes

- Transcoder fails to calculate PTS to DTS ratio (I/O error2 - 22) (#2636).
- Transcoder accessing files in parent folder (#2608).

### 18.24.3 4.5.7

*2016-12-21*

#### Transcoder fixes

- High memory usage of idle transcoder (mmap threshold) (#2569).
- Memory leak with multi-page images (#2573).
- Huge transcoder log files (#2570).

### 18.24.4 4.5.6

*2016-11-17*

#### Bug fixes

- Incorrect media type for ProTools (.ptx) files (#2558).
- NPE when setting indextimespans=false and using item field groups (#2545).

#### Transcoder fixes

- Transcoder updates modification time on input file (#2460).

### 18.24.5 4.5.5

*2016-10-13*

#### Bug fixes

- Export job does not properly save files from being deleted (#2510).
- Slow reindex of standalone files (#2507).
- Sidecar import of STL file fails "Failed to parse sidecar file: EOF" (#2506).
- Export job creates sidecar XML with extension .xml.xml when using filename script (#2473).
- Possible NPE when reindexing items with thumbnails on S3 (#2418).

#### Server fixes

- Config file /etc/sysconfig/vidispine and license replaced on upgrade (#2497).

#### Transcoder fixes

- D-10 VITC lines should be removed from poster (#2509).
- Transcoder does not handle content without aspect ration information (#2508).
- Don't create thumbnails for all layers in an image (#2472).

### 18.24.6 4.5.4

*2016-08-01*

### Bug fixes

- File copy failing between S3 and S3 compatible storages (#2389).
- NPE on container raw passkey import (#2433).
- Fix waiting raw passkey import job not aborting (#2342).

### Transcoder fixes

- Broken encoding of AAC in MP4 (#2425).

## 18.24.7 4.5.3

*2016-06-21*

### Bug fixes

- Notification without media type causing notifications to not be sent (#2384).
- Cannot connect to S3 endpoint that requires *v2 signatures* (#2383).
- Use the storage with more free capacity using the high watermark evacuation (#2359).
- Do not move file to a storage without enough free space duration high watermark evacuation (#2360).
- Thumbnails on S3 sometimes cannot be updated (#2386).
- Updating thumbnails on S3 causing item reindex to fail (#2385).

### Server fixes

- Index messages not rolled back on error (#2348).

### Transcoder fixes

- Accurate editing of audio-only MXF files (#2394).

## 18.24.8 4.5.2

*2016-05-09*

### Security notice

- Fix for CVE-2016-3714 (https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=2016-3714).

  Although it is not likely that Vidispine systems are affected, as Vidispine does its own file type checking, the updated transcoder contains the suggested fix to the ImageMagick policy.xml (#2331).

### Bug fixes

- PSQLException when deleting item with large metadata (#2330).
- Slow mime-type check causing steps to disappear and retry (#2332).
- NPE when retrieving empty access control list (#2288).
- Collection metadata update with skipForbidden=True allows update of read-only fields (#2340).
- Azure HTTP x-ms-version header error (#2338).
- Fix duplicate keys when batch deleting AWS SQS messages (#2312).
- Error showing collection access graph if collection has ancestors (#2286).

- HTTP 500 when fetching components using content path and query result is empty (#2337).
- Library empty if hits greater than number of returned items (#2301).
- HTTP 500 when fetching default values for item with field groups (#2336).
- No jobMetadata for thumbnail job (#2329).
- S3 storage handler cannot successfully refresh from buckets via Google Storage Interoperability (#2300).

### Server fixes

- Missing unique constraints on collection relationship tables (#2252).
- RPM upgrade recreates deployment License.lic which overrides slaveAuth.lic (#2307).
- Force server to use LC_ALL=en_US.UTF-8 (#2311).
- Adjust start and stop priority for vidispine service on CentOS (#2150).

### Transcoder fixes

- Correct shape deduction of MPEG transport streams with large frame sizes (#2309).
- Handle content where SPS/PPS does not match container timebase (#2335).
- Properly handle mixing of audio-only MXFs (#2281).
- For FLV files, read dimensions from container metadata instead of codec data (which is typically rounded up to the multiple of 8) (#2339).
- Fix for calculation of burnt-in timecode for QuickTime source sources contains EDLs (#2334).
- Properly handle MPEG-2 with separate field encoding (#2333).

  For MPEG-2 interlaced video where fields are encoded separately, and stored in MPEG transport stream, use the following setting in the transcode preset document:

```
<demuxerSetting>
  <key>pts_mode</key>
  <value>original</value>
</demuxerSetting>
```

  This will be the default in Vidispine 4.6.

## 18.24.9 4.5.1

*2016-03-29*

### MatrixStore

- The MatrixStore SDK has been updated from 3.1.3.3 to 3.1.3.7. This fixes an exception when reading XML files from MatrixStore (#2246).

### Bug fixes

- Solr self test fails when using SolrCloud (#1974).
- Incorrect Solr query when using nested OR operators (#2274).
- Jobs not running on MySQL (SQLGrammarException) (#2245).
- Slow update of user groups (#2272).

- Missing object metadata in MatrixStore (MXFS_FILENAME_UPPER) (#2271).
- SFTP file worker connection leak on file delete (#2249).
- IllegalArgumentException on item list to Azure (#2254).
- Moving a metadata timespan breaks references (#2218).
- Slow access DOT graph for items in a large amount of collections (#2251).
- No thumbnails created for certain files with an EDL (#2273).
- No document metadata for PDF files when transcoding on import (#2257).

#### Server fixes

- vidispine-admin failing if database URL contains port number (#2266).

#### Agent fixes

- Incorrect transcode output written to S3 and DS3 by VSA (#2276).
- Incorrect exit code from vidispine-agent-admin (#2267).

#### Transcoder fixes

- Added ability to use MainConcept MPEG-2 encoder (codec: `mc_mpeg2video`) for generic MPEG-2 for maximum control of encoding properties (#2279).
- Fixed problem canvas size when scaling images with non-proportional factors (#2268).
- Fix crash when using demuxerSetting and exception is thrown (#2282).
- PDF page count in shape metadata (image_pages) (#2283).
- Support *image sharpening* (#2278).
- Use DTS as fallback for missing PTS values (#2259).
- Long MediaInfo times for fragmented files (#2277).
- Handle WAV file with metadata of odd number of bytes (#2258).

### 18.24.10 4.5

*2016-02-22*

#### Vidispine agent with S3 compatible storages

The agent has been updated to support S3 compatible storages as shares. This allows on-premise storages and archives to be connected to VS via the agent (#2174).

#### Spectra Logic BlackPearl

Support for connecting Spectra Logic BlackPearl storage systems with Spectra S3 interface (#2049). Works for both Vidispine and Vidispine Agent. Connect as storage using the *ds3 scheme*. MD5, CRC32 and CRC32c checksums are supported.

#### Color profiles

Color profiles management has been improved for PDFs (#2242) and bitmap images (#2244). Color profiles can now be detected from XMP metadata. Several profiles are now bundled with the transcoder. See *Image-only settings*.

### Content paths and aliases

The item search resources have been updated with support for *content paths and aliases*. This allows for reduced response sizes and improved performance when fetching content relating to an item (#2133).

```
GET API/item?p=id,v(title),shape[tag=original].containerComponent.[format,file]
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <hits>108265</hits>
  <item id="VX-481945">
    <metadata>
      <timespan start="-INF" end="+INF">
        <field>
          <name>title</name>
          <value>Stand On The Rock</value></field>
        </timespan>
      </metadata>
    </item>
    <shape>
      <tag>original</tag>
      <containerComponent>
        <file>
          <id>VX-5121000</id>
          <path>VX-5121000.jpg</path>
          <uri>file:///srv/media/VX-5121000.jpg</uri>
          <state>CLOSED</state>
          <size>12495</size>
          <hash>3fd6295d57e921f87505acce885e716930943c9f</hash>
          <timestamp>2016-02-21T12:24:10.967+01:00</timestamp>
          <refreshFlag>1</refreshFlag>
          <storage>VX-1</storage>
        </file>
        <format>JPEG</format>
      </containerComponent>
    </shape>
  </item>
  ...
</ItemListDocument>
```

The *batch list job* and *list library job* has also been updated to support content paths.

### Search improvements

Full-text queries can now be used in search operators (#2126). This allows you to for example find collections matching one text query and that contain items matching another text query.

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <operator operation="OR">
    <operator operation="AND">
      <text>Time</text>
    </operator>
    <item>
      <operator operation="AND">
        <text>Sooner or later</text>
      </operator>
    </item>
  </operator>
</ItemSearchDocument>
```

When *searching for metadata field groups* you can now opt to search for field groups only from items, collections, document or the global metadata (#2156).

```
PUT /metadata-field/field-group?source=item
Content-Type: application/xml

<MetadataFieldGroupSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>app_comment</name>
    <value>Listen up</value>
  </field>
</MetadataFieldGroupSearchDocument>
```

The performance of metadata updates on systems containing a large number of references has also been improved (#2145).

### External identifiers

Collections and libraries now both support *external ids* (#1335).

### Collection metadata improvements

The *collection metadata* resource has been updated with a number of endpoints that were previously only available for the item metadata.

- It is now possible to view and manage change sets for a collection just as with an item (#2140).
- Metadata on collections can now also be moved and modified by UUID. (#2155).

### Export templates

Export templates are available as a beta version. The beta version means that syntax may change somewhat for the final implementation. This version is also a call for feedback for other functionality that you would like to see (#2117).

### Other improvements

- The Amazon S3 *infrequent access* storage class is now supported (#2184).
- Libraries can now be listed using the *list library resource*, making it possible to more easily list a specific set of items (#2139).
- The position and style of burnt in timecodes can now be *customized per preset* (#2104).
- External transcodes support using non-local watchfolders (#2240).
- Support for adding *custom JARs* with vidispine-server (#2208).

### Bug fixes

- Possible NPE when importing EVS sidecar XML (#2178).
- Deleting conflicting metadata in one request (#2118).

### Transcoder fixes

- Transcoder crash on moof files with traf without streams (#2243).
- Transcoder uses a new ImageMagick, that fixes color issues with certain PDF/PS.
- Transcoder exposes some HTTP buffer settings, see *HTTP buffer sizes*.

#### Deprecated in 4.5

Support for GlassFish is being phased out and will be removed in Vidispine 4.6. We recommend that you switch to *Installation* when upgrading to Vidispine 4.3 or later.

#### Removed in 4.5

Support for JBoss has been discontinued. Use *Installation* instead.

#### Upgrading from 4.4

- Solr: No changes to the documents. Re-indexing is not required.
- Support for JBoss has been discontinued. Use *Installation* instead.

## 18.25 Version 4.4

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.25.1 4.4.4

*2016-05-09*

#### Security notice

- Fix for CVE-2016-3714 (https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=2016-3714).

  Although it is not likely that Vidispine systems are affected, as Vidispine does its own file type checking, the updated transcoder contains the suggested fix to the ImageMagick policy.xml (#2331).

#### Bug fixes

- PSQLException when deleting item with large metadata (#2330).
- Slow mime-type check causing steps to disappear and retry (#2332).
- NPE when retrieving empty access control list (#2288).
- Collection metadata update with skipForbidden=True allows update of read-only fields (#2340).

#### Transcoder fixes

- Fix for calculation of burnt-in timecode for QuickTime source sources contains EDLs (#2334).
- Properly handle MPEG-2 with separate field encoding (#2333).

  For MPEG-2 interlaced video where fields are encoded separately, and stored in MPEG transport stream, use the following setting in the transcode preset document:

```
<demuxerSetting>
  <key>pts_mode</key>
  <value>original</value>
</demuxerSetting>
```

  This will be the default in Vidispine 4.6.

### 18.25.2 4.4.3

*2016-03-24*

#### MatrixStore

- The MatrixStore SDK has been updated from 3.1.3.3 to 3.1.3.7. This fixes an exception when reading XML files from MatrixStore (#2246).

#### Bug fixes

- Jobs not running on MySQL (SQLGrammarException) (#2245).
- Slow update of user groups (#2272).
- Missing object metadata in MatrixStore (MXFS_FILENAME_UPPER) (#2271).
- SFTP file worker connection leak on file delete (#2249).
- IllegalArgumentException on item list to Azure (#2254).
- Moving a metadata timespan breaks references (#2218).
- Slow access DOT graph for items in a large amount of collections (#2251).
- No thumbnails created for certain files with an EDL (#2273).
- No document metadata for PDF files when transcoding on import (#2257).

#### Server fixes

- vidispine-admin failing if database URL contains port number (#2266).

#### Agent fixes

- Incorrect exit code from vidispine-agent-admin (#2267).

#### Transcoder fixes

- Support *image sharpening* (#2278).
- Use DTS as fallback for missing PTS values (#2259).
- Long MediaInfo times for fragmented files (#2277).
- Handle WAV file with metadata of odd number of bytes (#2258).

### 18.25.3 4.4.2

*2016-02-22*

#### Bug fixes

- Slow listing of files on Azure (#2187).
- Video misidentified as image/cgm (#2185).
- Export transcode failure should fail export (#2183).
- File removed from Solr on shape delete with keepFiles=true (#2212).
- NPE on item reindex on thumbnail read error (#2221).
- Don't use IN subquery statements with MySQL (#2177).
- Be able to skip indexing of timed metadata (#2212).
- `POST /storage/(storage-id)/file/data` not creating file when using *segment files* (#2171).
- NPE instead of 404 for `API/storage/{external-id}/file/{id}` requests (#2179).

- NPE on DB migrate when correcting column nullable status (#2190).
- Files on local file system not found on Windows (#2238).
- Schema file transcoder.xsd missing from API (#2217).
- Make shape information available to conform presets (#2189).
- Temporary segment files not deleted (#2239).
- External transcode completing too early (#2241).
- Cannot update items' metadata via library and JSON (#2216).

### Server fixes

- APInoauth application.wadl missing from vidispine-server (#2214).

### Agent fixes

- Incorrect service status from vidispine-agent-admin on CentOS 6 (#2096).

### Transcoder fixes

- Insecure parsing of m3u4 files (#2222).
- Shape deduction crash on invalid QuickTime file (#2180).
- Make transcoder fail on transcode of truncated QuickTime files (#2220).
- Transcoder crash on very short clip (#2223).

## 18.25.4 4.4.1

*2015-12-16*

### Improvements

- Transcode jobs are now more resilient to connection errors. Use the `maxTranscoderUnavailableTime` property to control how long a transcoder may be unreachable before the job should fail (#2166).
- Use the `bulkyMetadataKeysToIgnore` property to avoid saving unused analysis results from analyze jobs in the bulky metadata (#2165).
- Support for marking a file as a `duplicate of another` (#2129).
- The *thumbnail format* used by the transcoder is now exposed as a setting (#2128).
- How frequently the transcode status in a job is updated can be configured using the `transcoderNonblockingStatusInterval` property (#2164).

### Amazon S3 improvements

- Use the *sseAlgorithm* parameter to set the `x-amz-server-side-encryption` header for S3 object upload requests. See Amazon S3 Server-Side Encryption (http://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html) for more information (#2103).

### Transcoder improvements

- The transcoder will do a further analysis of GOP timecodes in MPEG video. If the first GOPs are not contiguous, the startTimecode element in the videoComponent of the shape is not present (#2176).
- The transcoder updates the GOP timecodes in MPEG video according to the TranscodePresetDocument of the shape-tag. This functionality also works when only remuxing (#2141).
- Ability to control GOP structure based on scene changes, see *sceneChangeThreshold*. Mostly used to disable scene change detection (#2160).
- Ability to disable writing of time code track, see *noTimeCodeTrack* (#2143).

### Bug fixes

- No delay before job step retry if the asynchronous part of the step failed (#2161).
- The population of the *__representativeThumbnails* field would slow down collection metadata retrieval (#2119).
- Storage rule with a single `<not>` entry not honoured (#2158).
- All shape components not deleted on placeholder import failure (#2163).
- Cannot rename file on S3 (#2111).
- Shape created on wrong storage when doing *raw shape imports* and using the `tag` parameter (#2167).
- NPE when evaluating task groups for AUTO_IMPORT jobs (#2125).
- Signiant transfers failing for large files (#2152).
- Cannot delete group used in many ACLs on GlassFish (#2144).
- Username parameter ignored for *bulk item exports* (#2137).
- Rejected shape import requests causing empty jobs to be created (#2115).
- Configurable exclusive job step wait time. See *jobExclusiveStepMaxWait* (#2116).
- Export job failing to find and copy file created from transcode step (#2106).
- List item jobs not writing directly to S3 (#2120).
- List item jobs not finishing when listing by group (#2108).
- No thumbnails created for MTS files (#2107).

### Transcoder fixes

- Transcoder fails to read from HTTPS (#2162).
- Transcoder crash on aborted jobs (#2159).
- Transcoder stuck on broken PDF (#2122).
- Add option to fail if frame cannot be decoded (#2124).
- Incorrect scaling of posters (#2123).

## 18.25.5 4.4

*2015-10-23*

### Vidispine server agent

Support for the Vidispine Server Agent (VSA), a bundle of an agent for on-premise file storages and the Vidispine transcoder (#1806).

- Enables cloud deployments of Vidispine while keeping essence in the original place.
- VSA is available both for Ubuntu/Debian and CentOS/RedHat releases. See *Vidispine Server Agent* on how to install the VSA packages and configuration.

### Amazon S3 improvements

- Amazon S3 file notifications are now supported, making it possible to have Amazon push file event notifications to Vidispine instead of having Vidispine poll buckets on S3 (#1949).

  See *S3 Event SQS Notifications* on how to set up the Amazon SQS (https://aws.amazon.com/sqs/) queue, configure S3 event notifications and connect the queue to a storage.
- Buckets in Beijing and Frankfurt are now supported as the latest AWS SDK is now in use which supports the signature version 4 signing process (#1712).
- The *region and endpoint* can be specified when configuring a storage. This is required when using buckets in the two new regions (#2040).

### Transcoder discovery

Transcoders can be automatically *discovered* via either DNS or HTTP. The former allows integration with DNS servers and services such as Amazon Route 53 (http://aws.amazon.com/route53/) and Consul (http://consul.io/) (#2092).

```
<?xml version="1.0"?>
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <id>VX-1</id>
  <transcoder>
    <type>DIRECTORY</type>
    <url>dns://consul.example.com/transcoders.example.com</url>
    <state>ONLINE</state>
    <transcoder>
      <url>http://t1.transcoder.example.com:8888/</url>
      <version>4.4</version>
      <state>ONLINE</state>
    </transcoder>
    <transcoder>
      <url>http://t2.transcoder.example.com:8888/</url>
      <state>OFFLINE</state>
    </transcoder>
  </transcoder>
</ResourceDocument>
```

### Transcoder sharing

Unexpected transcoder license errors have been an issue in the past when one or more Vidispine instances have been connected to the same transcoder. This is no longer an issue as the transcoder will respect the license of each instance (#2051).

See *Using multiple transcoders* on how to set this up.

### Task groups

*Task groups* are a new entity that pairs an entity with a resource according to some rule. In this release these can be used to control the transcoders that should be used for certain types of jobs, for example to enforce that R3D jobs must

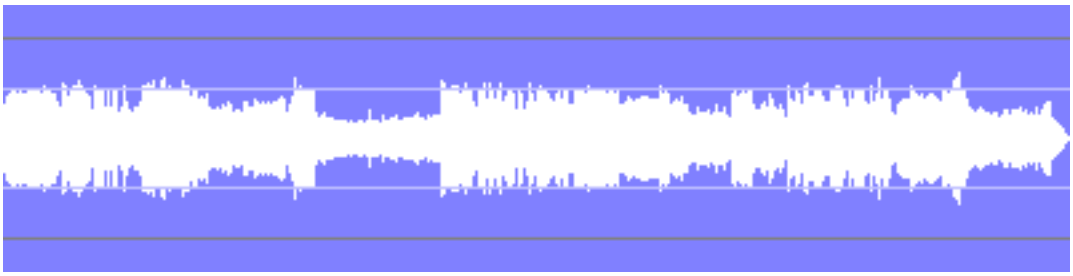
use the transcoder supporting hardware decode of R3D, or to reserve some transcoders for high priority jobs (#2094).

```
<TaskGroupDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <job>
    <priority>HIGH</priority>
  </job>
  <transcoder id="VX-1"/>
  <transcoder id="VX-2"/>
  <priority>MEDIUM</priority>
</TaskGroupDocument>
```

### Audio waveforms

The *Shape analysis* now also store information that can be used to create an "audio waveform". After the analysis has been made, the waveform information is available either as numerical values or image (PNG, one image per channel) (#2039).

- See *Waveform information* for how to retrieve the waveform information.



### Other features

- Support for *OAuth 2.0 bearer-token authentication* (#1918).
- More efficient online check of S3 buckets(#2073).
- Ability to analyze a shape with multiple files (#2020).
- Ability to transcode directly into another item (#2044).
- Ability to add multiple overlays in same transcode profile (#2061).

### Bug fixes

- Not able to proxy files in sequence render jobs (#2026).
- Conform job doesn't support audio mappings in preset (#2033).
- Exponential backoff for repeatedly failing services to prevent exhaustive logging (#2070).
- Deadlock in jobs would cause the job executor to shut down and prevent job execution until job gets marked as DISAPPEARED and later restarted (#2067).
- FTP storage to empty directory does not work (#2062).

### Transcoder fixes

- Media check for .m2v files is time consuming (#2066).
- Transcoding from 23.976 FPS transport streams renders output out of sync (#2064).

### Upgrading from 4.3

- Solr: No changes to the documents. Re-indexing is not required.
- The property `indexCollectionItemOrder` is now set to `false` by default. Set it to `true` before upgrading if you rely on the old behaviour.
- The transcode preset script is now also evaluated for conform jobs. The difference compared to transcodes is that the shape is empty. If you use the same preset for both transcodes and conforms, then make sure that the script verifies the existence of the components of the shape before using them to avoid null pointer exceptions from the script.

## 18.26 Version 4.3

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.26.1 4.3.8

*2016-05-09*

#### Security notice

- Fix for CVE-2016-3714 (https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=2016-3714).

  Although it is not likely that Vidispine systems are affected, as Vidispine does its own file type checking, the updated transcoder contains the suggested fix to the ImageMagick policy.xml (#2331).

#### Bug fixes

- PSQLException when deleting item with large metadata (#2330).
- Slow mime-type check causing steps to disappear and retry (#2332).
- NPE when retrieving empty access control list (#2288).
- Collection metadata update with skipForbidden=True allows update of read-only fields (#2340).

#### Transcoder fixes

- Fix for calculation of burnt-in timecode for QuickTime source sources contains EDLs (#2334).
- Properly handle MPEG-2 with separate field encoding (#2333).

  For MPEG-2 interlaced video where fields are encoded separately, and stored in MPEG transport stream, use the following setting in the transcode preset document:

```
<demuxerSetting>
  <key>pts_mode</key>
  <value>original</value>
</demuxerSetting>
```

  This will be the default in Vidispine 4.6.

### 18.26.2 4.3.7

*2016-03-22*

#### MatrixStore

- The MatrixStore SDK has been updated from 3.1.3.3 to 3.1.3.7. This fixes an exception when reading XML files from MatrixStore (#2246).

#### Bug fixes

- Jobs not running on MySQL (SQLGrammarException) (#2245).
- Slow update of user groups (#2272).
- Missing object metadata in MatrixStore (MXFS_FILENAME_UPPER) (#2271).
- SFTP file worker connection leak on file delete (#2249).
- IllegalArgumentException on item list to Azure (#2254).
- Moving a metadata timespan breaks references (#2218).
- Slow access DOT graph for items in a large amount of collections (#2251).
- No thumbnails created for certain files with an EDL (#2273).
- No document metadata for PDF files when transcoding on import (#2257).

#### Server fixes

- vidispine-admin failing if database URL contains port number (#2266).

#### Transcoder fixes

- Long MediaInfo times for fragmented files (#2277).
- Handle WAV file with metadata of odd number of bytes (#2258).

### 18.26.3 4.3.6

*2016-02-22*

#### Bug fixes

- Slow listing of files on Azure (#2187).
- Video misidentified as image/cgm (#2185).
- Export transcode failure should fail export (#2183).
- File removed from Solr on shape delete with keepFiles=true (#2212).
- NPE on item reindex on thumbnail read error (#2221).
- Don't use IN subquery statements with MySQL (#2177).
- Be able to skip indexing of timed metadata (#2212).
- `POST /storage/(storage-id)/file/data` not creating file when using *segment files* (#2171).
- NPE instead of 404 for `API/storage/{external-id}/file/{id}` requests (#2179).

### Transcoder fixes

- Insecure parsing of m3u4 files (#2222).
- Shape deduction crash on invalid QuickTime file (#2180).
- Make transcoder fail on transcode of truncated QuickTime files (#2220).
- Transcoder crash on very short clip (#2223).

## 18.26.4 4.3.5

*2015-12-07*

### Improvements

- Support for marking a file as a `duplicate of another` (#2129).
- The *thumbnail format* used by the transcoder is now exposed as a setting (#2128).

### Bug fixes

- Signiant transfers failing for large files (#2152).
- Cannot delete group used in many ACLs on GlassFish (#2144).
- Username parameter ignored for *`bulk item exports`* (#2137).

### Transcoder fixes

- Transcoder crash on aborted jobs (#2159).

## 18.26.5 4.3.4

*2015-11-06*

### Bug fixes

- Deadlock on job start causing job to stop running (#2067).
- Rejected shape import requests causing empty jobs to be created (#2115).
- Configurable exclusive job step wait time. See *`jobExclusiveStepMaxWait`* (#2116).
- Export job failing to find and copy file created from transcode step (#2106).
- List item jobs not writing directly to S3 (#2120).
- List item jobs not finishing when listing by group (#2108).
- FTP storage to empty directory does not work (#2062).
- No thumbnails created for MTS files (#2107).

### Transcoder fixes

- Transcoder stuck on broken PDF (#2122).
- Add option to fail if frame cannot be decoded (#2124).
- Incorrect scaling of posters (#2123).

### 18.26.6 4.3.3

*2015-10-01*

#### Bug fixes

- Encrypt credentials in thumbnail resource URIs (#1982).
- Storage rule supervisor repeatedly restarting due to a long running transaction, causing copy jobs not to start (#2050).
- High JMS connection usage from storage supervisor (#2047).
- Slow file system metadata updates causing growing imports to end early (Tunable growing file timeout) (#2053).
- Slow library search due to rollbacks (#2028).
- Cached search bigtext entries not always removed (#2041).
- Solr searchers keeping GlassFish from shutting down (#2038).
- Quote characters in a collection name generates broken access DOT graph (#2052).
- Checksum not computed for file with %2B in name (#2056).

#### Server fixes

- Checksum fails to compute for large files with vidispine-server (#2022).
- No version available from vidispine-server –version (#2034).
- Don't put expired messages on the DLQ (#2023).

#### Transcoder fixes

- Interlace flag not working for H.264 (#2036).
- Incorrect subclipping of MXF OP1a with index in footer (#2057).
- Incorrect bit depth for AES3 (#2059).
- 32bit lpcm detected as 16bit (#2058).
- Burnt in timecode text offset from text box (#2060).

### 18.26.7 4.3.2

These are release highlights. For a full listing of the features, bug fixes and upgrade notes, please see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

*2015-08-27*

#### Improvements

- New Relic users will now see web requests with transaction names that match the path used in the request, instead of the automatic name assigned by New Relic which tends to group requests to different endpoints together.
- Speed ups when faceting on many fields with many matching terms.

### Bug fixes

- Cached search documents not being removed from the big text table, causing it to grow over time.
- The periodic update of a large library could cause library API requests and index updates to halt due to a locking issue.
- When using SolrCloud the connection manager could abort long running requests, typically causing updates to Solr to fail.
- Various bug fixes and improvements.

For the full list of changes, see the release notes (http://www.vidispine.com/partner/my-documentation).

## 18.26.8 4.3.1

These are release highlights. For a full listing of the features, bug fixes and upgrade notes, please see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

*2015-07-03*

### RHEL 7 support

Red Hat Enterprise Linux 7 is now a supported. You may have already seen some EL7 RPMs in 4.3, but now the packaging of the transcoder has also been completed making EL7 the latest addition to the list of supported operating system.

### System overview

The metrics page available on the admin port when running standalone Vidispine has been redesigned. It now also features graphs of some of the metrics exposed via StatsD and the metrics admin resource.

### Other

- Fixed incorrect role on `PUT /collection/(collection-id)/metadata`.
- JavaScript notifications now work on MySQL.
- WAITING jobs could cause job metrics to become incorrect.
- Various bug fixes and improvements.

For the full list of changes, see the release notes (http://www.vidispine.com/partner/my-documentation).

## 18.26.9 4.3

These are release highlights. For a full listing of the features, bug fixes and upgrade notes, please see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

*2015-06-02*

### Standalone deployment

Vidispine can now be run without the need of an application server. See *Installation*.

### Group search

It is now possible to *query field groups* when searching for items or collections.

```
<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <group>
    <name>movie_info</name>
    <field>
      <name>movie_name</name>
      <value>StarWars</value>
    </field>
    <field>
      <name>episode_no</name>
      <value>1</value>
    </field>
  </group>
</ItemSearchDocument>
```

#### Other

- Support for *automatic removal* of old jobs.

## 18.27 Version 4.2

The release notes will tell you what's new in each version, and any changes that you must be aware of when upgrading. For reference, Vidispine ticket numbers are printed as (#1234).

### 18.27.1 4.2.16

*2016-08-31*

#### Bug fixes

- Race between copy job and storage worker causing job to fail (#2462).
- Demuxer settings not passed to transcoder for thumbnail jobs (#2461).

#### Transcoder fixes

- D10 blanking lines included in posters (#2463).
- Incorrect scaling of posters from NTSC files (#2464).

### 18.27.2 4.2.15

*2016-05-09*

#### Security notice

- Fix for CVE-2016-3714 (https://www.cve.mitre.org/cgi-bin/cvename.cgi?name=2016-3714).

  Although it is not likely that Vidispine systems are affected, as Vidispine does its own file type checking, the updated transcoder contains the suggested fix to the ImageMagick policy.xml (#2331).

#### Bug fixes

- PSQLException when deleting item with large metadata (#2330).
- Slow mime-type check causing steps to disappear and retry (#2332).
- NPE when retrieving empty access control list (#2288).

- Collection metadata update with skipForbidden=True allows update of read-only fields (#2340).

### Transcoder fixes

- Fix for calculation of burnt-in timecode for QuickTime source sources contains EDLs (#2334).
- Properly handle MPEG-2 with separate field encoding (#2333).

  For MPEG-2 interlaced video where fields are encoded separately, and stored in MPEG transport stream, use the following setting in the transcode preset document:

```
<demuxerSetting>
  <key>pts_mode</key>
  <value>original</value>
</demuxerSetting>
```

  This will be the default in Vidispine 4.6.

## 18.27.3 4.2.14

*2016-03-21*

### MatrixStore

- The MatrixStore SDK has been updated from 3.1.3.3 to 3.1.3.7. This fixes an exception when reading XML files from MatrixStore (#2246).

### Bug fixes

- Jobs not running on MySQL (SQLGrammarException) (#2245).
- Slow update of user groups (#2272).
- Missing object metadata in MatrixStore (MXFS_FILENAME_UPPER) (#2271).
- SFTP file worker connection leak on file delete (#2249).
- IllegalArgumentException on item list to Azure (#2254).
- Moving a metadata timespan breaks references (#2218).
- Slow access DOT graph for items in a large amount of collections (#2251).
- No thumbnails created for certain files with an EDL (#2273).
- No document metadata for PDF files when transcoding on import (#2257).

### Transcoder fixes

- Handle WAV file with metadata of odd number of bytes (#2258).

## 18.27.4 4.2.13

*2016-02-16*

### Bug fixes

- Slow listing of files on Azure (#2187).
- Video misidentified as image/cgm (#2185).
- Export transcode failure should fail export (#2183).

- File removed from Solr on shape delete with keepFiles=true (#2212).
- NPE on item reindex on thumbnail read error (#2221).
- Don't use IN subquery statements with MySQL (#2177).
- Be able to skip indexing of timed metadata (#2212).
- `POST /storage/(storage-id)/file/data` not creating file when using *segment files* (#2171).
- NPE instead of 404 for `API/storage/{external-id}/file/{id}` requests (#2179).

### Transcoder fixes

- Insecure parsing of m3u4 files (#2222).
- Shape deduction crash on invalid QuickTime file (#2180).
- Make transcoder fail on transcode of truncated QuickTime files (#2220).
- Transcoder crash on very short clip (#2223).

## 18.27.5 4.2.12

*2015-12-07*

### Improvements

- Support for marking a file as a `duplicate of another` (#2129).
- The *thumbnail format* used by the transcoder is now exposed as a setting (#2128).

### Bug fixes

- Signiant transfers failing for large files (#2152).
- Cannot delete group used in many ACLs on GlassFish (#2144).
- Username parameter ignored for *`bulk item exports`* (#2137).

### Transcoder fixes

- Transcoder crash on aborted jobs (#2159).

## 18.27.6 4.2.11

*2015-11-06*

### Bug fixes

- Deadlock on job start causing job to stop running (#2067).
- Rejected shape import requests causing empty jobs to be created (#2115).
- Configurable exclusive job step wait time. See *`jobExclusiveStepMaxWait`* (#2116).
- Export job failing to find and copy file created from transcode step (#2106).
- List item jobs not writing directly to S3 (#2120).
- List item jobs not finishing when listing by group (#2108).
- FTP storage to empty directory does not work (#2062).

### Transcoder fixes

- Transcoder stuck on broken PDF (#2122).
- Add option to fail if frame cannot be decoded (#2124).
- Incorrect scaling of posters (#2123).

## 18.27.7 4.2.10

*2015-10-01*

### Bug fixes

- Storage rule supervisor repeatedly restarting due to a long running transaction, causing copy jobs not to start (#2050).
- High JMS connection usage from storage supervisor (#2047).
- Slow file system metadata updates causing growing imports to end early (Tunable growing file timeout) (#2053).
- Slow library search due to rollbacks (#2028).
- Cached search bigtext entries not always removed (#2041).
- Solr searchers keeping GlassFish from shutting down (#2038).
- Quote characters in a collection name generates broken access DOT graph (#2052).
- Checksum not computed for file with %2B in name (#2056).

### Transcoder fixes

- Interlace flag not working for H.264 (#2036).
- Incorrect subclipping of MXF OP1a with index in footer (#2057).
- Incorrect bit depth for AES3 (#2059).
- 32bit lpcm detected as 16bit (#2058).
- Burnt in timecode text offset from text box (#2060).

## 18.27.8 4.2.9

> These are release highlights. For a full listing of the features, bug fixes and upgrade notes, please see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

*2015-08-27*

This is a bug fix release with a large number of fixes.

### Improvements

- New Relic users will now see web requests with transaction names that match the path used in the request, instead of the automatic name assigned by New Relic which tends to group requests to different endpoints together.
- Speed ups when faceting on many fields with many matching terms.

### Bug fixes

- Cached search documents not being removed from the big text table, causing it to grow over time.
- The periodic update of a large library could cause library API requests and index updates to halt due to a locking issue.
- When using SolrCloud the connection manager could abort long running requests, typically causing updates to Solr to fail.
- Various bug fixes and improvements.

For the full list of changes, see the release notes (http://www.vidispine.com/partner/my-documentation).

## 18.27.9 4.2.8

These are release highlights. For a full listing of the features, bug fixes and upgrade notes, please see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

*2015-07-03*

This is a minor bug fix release that also adds two new job parameters.

### Improvements

New job parameters:

- `cerifyPriority` - To set the *priority of jobs in Cerify*.
- `checksumMode` - To have checksums for imported items computed during the transfer step. See *Checksum on file transfer*.

### Bug fixes

- Storage not marked as offline if MatrixStore vault goes offline.
- Poster format settings not working.
- Installer ignoring required but failing HTTP requests.
- Certain transcodes/render jobs crashing the transcoder.
- Various bug fixes and improvements.

For the full list of changes, see the release notes (http://www.vidispine.com/partner/my-documentation).

## 18.27.10 4.2.7

These are release highlights. For a full listing of the features, bug fixes and upgrade notes, please see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

*2015-06-02*

This is a minor bug fix release. See the release notes (http://www.vidispine.com/partner/my-documentation) for details.

## 18.27.11 4.2.6

These are release highlights. For a full listing of the features, bug fixes and upgrade notes, please see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

*2015-05-13*

### Security notice

Apache POI (https://poi.apache.org/) has been updated from version 3.8 to 3.11 (https://poi.apache.org/changes.html) to mitigate XXE (https://www.owasp.org/index.php/XML_External_Entity_%28XXE%29_Processing) vulnerabilities (CVE-2014-3529 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3529) and CVE-2014-3574 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3574)) when extracting metadata from Office documents. This metadata extraction is only done if enabled using `parseFileMetadata`.

XXE has also been disabled in the XML parser used by Vidispine, to address vulnerabilities when parsing XML API requests and sidecar files for example.

### Performance improvements

This release contains a number of performance improvements.

For the API:

- Faster collection update and delete.
- Faster group update and delete.
- Faster merged access retrieval.
- Faster storage-rule creation.

For items:

- Faster poster generation.

### Changed defaults

Multi-site processing is now disabled by default. See `disableSiteCrunching`.

### Other

- Indexing of items with large metadata text fields could cause indexing to halt. This has now been fixed.
- Field groups in the item metadata are now indexed less often.
- Additional audio and video settings from the transcode preset are now supported when rendering or conforming.
- More control over burnt-in subtitle placements. See *Subtitle metadata fields and groups*.
- Various bug fixes and improvements.

## 18.27.12 4.2.5

> These are release highlights. For a full listing of the features, bug fixes and upgrade notes, please see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

*2015-02-26*

### Temporary transcoder path

It is now possible to use another directory for *temporary files for the transcoder*. This is controlled via the `<tempPath>` element in the transcoder. This element is controlled via *local configuration file* or *transcoder resource definition* or by changing the configuration for *all transcoders*.

### Optional hit count

The *hit count* can now be omitted from the results when searching. This can be done to reduce the response time of search requests.

```
GET API/item?count=false
```

### Other

- Buckets with files in subfolders are now properly scanned and will now appear on your S3 storages.
- *Proxy services* can be used for cloud licensing.
- A number of issues with the multi-site sync have been fixed.
- Various bug fixes and improvements.

## 18.27.13 4.2.4

These are release highlights. For a full listing of the features, bug fixes and upgrade notes, please see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

*2015-01-29*

### FTP connection pooling

If you perform a large number of imports or exports over high latency FTP connections then you can now create a *FTP connection pool* to reduce the overhead of establishing a new connection each time.

```
PUT /configuration/ftp-pool
Content-Type: application/xml

<FtpPoolConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <pool>
    <minSize>0</minSize>
    <maxSize>-1</maxSize>
    <evictionInterval>30000</evictionInterval>
    <minIdleTime>60000</minIdleTime>
  </pool>
<FtpPoolConfigurationDocument/>
```

### Find collections with specific items

An `item` subquery can now be used when searching for collections to find collections based on the items that they contain.

```
PUT /collection
Content-Type: application/xml

<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <operator operation="OR">
    <field>
      <name>title</name>
      <value>Peach</value>
    </field>
    <item>
      <field>
        <name>title</name>
        <value>Peach</value>
```

```
      </field>
    </item>
  </operator>
</ItemSearchDocument>
```

See *Searching for collections with specific items*.

### Transcoder transfer and hash computation

Available transcoder with direct file access to media can be used to compute hash sums of files, or to do filesystem-to-filesystem transfers. This will offload the application server.

To enable this, use the configuration variables `enableTranscoderHashing` and `enableTranscoderTransfer`.

### Database purging

Automatic routines for trimming two of the largest Vidispine tables are now included. For information about how to enable this, see *Database purging*.

## 18.27.14 4.2.3

> These are release highlights. For a full listing of the features, bug fixes and upgrade notes, please see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

*2014-12-18*

### StatsD metrics

Metrics related to operations performed by both Vidispine and the transcoders can now be exposed using the *StatsD* protocol. It is also possible to access the Vidispine metric statistics using *JMX*.

```
$ node stats.js config.js
17 Dec 12:23:31 - reading config file: config.js
17 Dec 12:23:31 - server is up
17 Dec 12:23:31 - DEBUG: Loading backend: ./backends/graphite
17 Dec 12:23:34 - DEBUG: vs.job.total.aborted:99|g
17 Dec 12:23:34 - DEBUG: vs.job.total.aborted_pending:0|g
17 Dec 12:23:34 - DEBUG: vs.job.total.failed_total:520|g
17 Dec 12:23:34 - DEBUG: vs.job.total.finished:50491|g
17 Dec 12:23:34 - DEBUG: vs.job.total.finished_warning:3|g
17 Dec 12:23:34 - DEBUG: vs.job.total.ready:0|g
17 Dec 12:23:34 - DEBUG: vs.job.total.started:1|g
17 Dec 12:23:34 - DEBUG: vs.job.total.waiting:0|g
17 Dec 12:23:34 - DEBUG: vs.service.load.5:0.02|g
17 Dec 12:23:34 - DEBUG: vs.service.load.60:0.02|g
...
```

### Multithreaded transcoder pipeline

The decoding part of the transcoder is now multithreaded for I-frame-only content. This means that for content such as ProRes, D10/IMX, etc, you will see a speed-up, especially if the input material is in high resolution.

### FileCatalyst transfers

FileCatalyst is now available as a transfer method between storage locations. The Vidispine application acts as a FileCatalyst client which can communicate to FileCatalyst servers for transferring files.

In order to register a FileCatalyst server for a storage, add a new transfer method to the storage.

```
<method>
  <uri>filecatalyst://fc:s3cret@localhost:2100/incoming/</uri>
  <type>TRANSFER</type>
</method>
```

For more information, see *FileCatalyst Integration*.

### StorNext file information

For files residing on a Quantum StorNext file system, Vidispine can now show metadata on the file. In order to use this, enable web services on StorNext and add a HSM method to the storage.

```
<method>
  <uri>stornext://webservice:webservice@localhost:81/stornext/snfs/</uri>
  <type>HSM</type>
</method>
```

For more information, see *StorNext Integration*.

### Standalone metadata

It is now possible to create *standalone documents* with arbitrary metadata. If you are using *global metadata* but need to store a large amount of data, leading to a large metadata documents, then consider splitting it up into smaller documents, for example by entity or group of entities.

```
PUT API/document/company_a
```

```
<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <timespan start="-INF" end="+INF">
    ...
  </timespan>
</MetadataDocument>
```

### Filters and facets

Search *filters* have been added to replace the facet filters. They support arbitrary queries compared to facet filters that only allow a single field to be queried.

Filters can also be excluded from certain facets. This can be used to reduce the number of search requests need to display search pages that uses multiple facets and multiple drill down options.

```
PUT API/item
Content-Type: application/xml

<ItemSearchDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <filter name="typeFilter">
    <field>
      <name>mediaType</name>
      <value>audio</value>
    </field>
  </filter>
  <facet count="true">
    <field>mediaType</field>
    <exclude>typeFilter</exclude>
  </facet>
</ItemSearchDocument>
```

```
<ItemListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <hits>1</hits>
  <item end="+INF" id="VX-361763" start="-INF">
    <timespan end="+INF" start="-INF"/>
  </item>
  <facet>
    <field>mediaType</field>
    <count fieldValue="none">1867</count>
    <count fieldValue="image">33</count>
    <count fieldValue="video">10</count>
    <count fieldValue="audio">1</count>
    <count fieldValue="data">1</count>
  </facet>
</ItemListDocument>
```

### Other

- It is no longer necessary to use the application server's `/tmp` directory to store output files for Azure, S3, FTP destinations. Instead, this can be handled using *segment files* on the destination storage.
- The timestamp handling for generating MP4/H.264 files have been rewritten. This means that proxy files are frame accurate without without the previous tweaks (`useDTSmode` et al).

For the full list of changes, see the release notes (http://www.vidispine.com/partner/my-documentation).

## 18.27.15 4.2.2

These are release highlights. For a full listing of the features, bug fixes and upgrade notes, please see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

*2014-11-03*

### Thumbnails on cloud storages

Thumbnails can now be stored on cloud storages such as Amazon S3 and Azure. Thumbnails will be stored using *one file per thumbnail*.

```
<ResourceDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <thumbnail>
    <path>direct+azure://:+kLZrqckLZrqckLZrqckLZrqc==@mystorage/my-container/</path>
  </thumbnail>
</ResourceDocument>
```

### Job pools

Use *job pools* to ensure that long running low priority jobs don't block high priority jobs from running.

```
PUT /configuration/job-pool
Content-Type: application/xml

<JobPoolListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <pool>
    <priorityThreshold>HIGH</priorityThreshold>
    <size>2</size>
  </pool>
  <pool>
    <priorityThreshold>LOWEST</priorityThreshold>
    <size>3</size>
```

```
  </pool>
</JobPoolListDocument>
```

### Shape and file search

It is now possible to search for shapes and files, based on their key-value metadata, using the *shape search* and the *file search* resources.

```
PUT /search/shape
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ShapeSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <field>
    <name>language</name>
    <value>en</value>
  </field>
</ShapeSearchDocument>
```

### Joins

Support for *joins* has also been added to allow for cross-entity search between items, shapes and files.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ItemSearchDocument version="2" xmlns="http://xml.vidispine.com/schema/vidispine">
  <text>peach</text>
  <shape>
    <field>
      <name>language</name>
      <value>en</value>
    </field>
  </shape>
</ItemSearchDocument>
```

### Platform

This release brings support for:

- PostgreSQL 9.3.
- Java 7 update 67.

### Other

- *Signiant* can now be used to transfer files between storages.

For the full list of changes, see the release notes (http://www.vidispine.com/partner/my-documentation).

## 18.27.16 4.2.1

These are release highlights. For a full listing of the features, bug fixes and upgrade notes, please see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

*2014-09-16*

### WADL improvements

A number of improvements have been made to the WADL file. Missing parameters have been added and duplicate parameters have been removed for example. It has been updated to also include:

- Parameter options.
- Markers for repeating parameters.

The WADL file can be obtained using `GET API/application.wadl`.

### Other

- Support for copying of DTV 608/708 Closed Captions.
- Files can now also be sorted by *file extension*.

For the full list of changes, see the release notes (http://www.vidispine.com/partner/my-documentation).

## 18.27.17 4.2

> These are release highlights. For a full listing of the features, bug fixes and upgrade notes, please see the release notes (http://www.vidispine.com/partner/my-documentation) in the Vidispine Partner Portal.

*2014-07-04*

### API documentation

The API documentation has been moved from the Vidispine wiki into this documentation that you are now reading.

It is available online at http://apidoc.vidispine.com/latest/ and also on your local installation at `/APIdoc`.

### Java 7

The 4.2 series now requires Java 7, specifically Java 7 update 25 as later versions have known bugs with GlassFish 3.x.

### Efficient file I/O

The new Java 7 File API is used to reduce the number of file system operations that are used when scanning local storages.

### Indexing

The Solr configuration now exists in the *indexing configuration*, but note that the Solr configuration properties are still supported.

This configuration can also be used to specify which fields should be included in the full text index, unless specified explicitly for a specific field.

```
PUT /configuration/indexing
Content-Type: application/xml

<IndexingConfigurationDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <solrPath>http://localhost:8088/solr</solrPath>
  <fieldDefault>
    <name>xmp_*</name>
    <fullText>false</fullText>
  </fieldDefault>
</IndexingConfigurationDocument>
```

### Detect renamed files

Renamed files can be detected and and re-associate based on the file checksum. Enable it using the `detectRenamedFiles` storage property.

### Platform

This release adds support for Ubuntu 14.04, Windows 2012 R2 and MySQL 5.6. At the same time, support for PostgreSQL 8.x, MySQL 5.1 and Java 1.6 has been discontinued.

### Other

- Ability to move fields from subgroups using *schema migrations*.
- Posters can now be created in either *JPEG or PNG format*.
- Support for storage exclude filters to exclude certain files from a storage. See `excludeFilter`.
- RED GPU acceleration
- Configurable *quality level* for RED file decoding.

For the full list of changes, see the release notes (http://www.vidispine.com/partner/my-documentation).

Other sources of information

- Partner portal (http://www.vidispine.com/partner/partnerportal), including the
  - latest versions of the software (https://repo.vidispine.com/),
  - knowledge base (http://www.vidispine.com/partner/knowledge-forum-support),
  - getting started guide (https://support.vidispine.com/space/CKB/650510858/Getting+Started+With+the+Vidispine+API+(Sta
    and
  - discussion forum

Other tools already available on your installation

- Selftest
- Log report
- XML Schemas (XSDs)
  - `xmlSchema.xsd`
  - `transcoder.xsd`
  - `common.xsd`
- Javadoc of compiled versions of the XML Schemas, useful when writing JavaScript integration code
- Python Client Library (https://pypi.org/project/vidispine/) our Vidispine Python Client Library available via PyPI

This copy of the documentation covers Vidispine version **21.3** (build **21.3-gdc92f56c3d-12409**). You can find other versions in the Partner Portal.

## /APInoauth

GET /APInoauth/callback/cloudconvert, 599
PUT /APInoauth/debug/echo, 598
POST /APInoauth/import/raw, 420
PUT /APInoauth/license/auth-info, 530
GET /APInoauth/selftest, 666
GET /APInoauth/stitch, 597
GET /APInoauth/vidispine-logs/job/(job-id).zip, 739

## /analyze-preset

GET /analyze-preset, **??**
GET /analyze-preset/(name), **??**
PUT /analyze-preset/(name), **??**
DELETE /analyze-preset/(name), **??**

## /application.wadl

GET /application.wadl, 599

## /auto-projection

GET /auto-projection, 530
GET /auto-projection/(name), 531
PUT /auto-projection/(name), 531
DELETE /auto-projection/(name), 532
PUT /auto-projection/(name)/disable, 532
PUT /auto-projection/(name)/enable, 532
GET /auto-projection/disable, 530
GET /auto-projection/enable, 530

## /collection

GET /collection, 356
PUT /collection, 378
POST /collection, 359
DELETE /collection, 360
GET /collection/(collection-id), 360
PUT /collection/(collection-id), 372
DELETE /collection/(collection-id), 359
PUT /collection/(collection-id)/(id), 371
DELETE /collection/(collection-id)/(id), 372
GET /collection/(collection-id)/ancestor, 367
GET /collection/(collection-id)/collection, 381
PUT /collection/(collection-id)/collection, 383
POST /collection/(collection-id)/export, 436
PUT /collection/(collection-id)/folder-name, 388
GET /collection/(collection-id)/item, 362
PUT /collection/(collection-id)/item, 367
POST /collection/(collection-id)/item/list, 366
PUT /collection/(collection-id)/map-to-folder, 387
DELETE /collection/(collection-id)/map-to-folder, 387
POST /collection/(collection-id)/order, 385
PUT /collection/(collection-id)/rename, 360
POST /collection/(collection-id)/train, 388
GET /collection/(id)/definition, 638
GET /collection/(id)/definition/(format), 638
PUT /collection/(id)/definition/(format), 638
DELETE /collection/(id)/definition/(format), 638
GET /collection/(id)/definition/(format)/asset, 639
PUT /collection/(id)/definition/(format)/asset, 639
DELETE /collection/(id)/definition/(format)/asset, 639
GET /collection/(id)/definition/(format)/extradata, 640
PUT /collection/(id)/definition/(format)/extradata,

640
DELETE /collection/(id)/definition/(format)/extradata, 640
POST /collection/(id)/version, 637
GET /collection/(id)/version/export, 646
POST /collection/(id)/version/export, 644
GET /collection/history, 381
POST /collection/project, 636
POST /collection/project/inspect, 640

## /configuration

GET /configuration, 392
GET /configuration/auth, 400
PUT /configuration/auth, 400
DELETE /configuration/auth, 400
GET /configuration/bulkymetadata, 400
PUT /configuration/bulkymetadata, 401
GET /configuration/cors, 397
PUT /configuration/cors, 398
GET /configuration/ftp-pool, 396
PUT /configuration/ftp-pool, 396
DELETE /configuration/ftp-pool, 396
GET /configuration/indexing, 392
PUT /configuration/indexing, 392
GET /configuration/job-pool, 394
PUT /configuration/job-pool, 394
DELETE /configuration/job-pool, 394
DELETE /configuration/job-pool/(priority), 395
GET /configuration/job-priority, 399
PUT /configuration/job-priority, 399
DELETE /configuration/job-priority, 399
GET /configuration/logreport, 397
PUT /configuration/logreport, 397
GET /configuration/metrics, 392
PUT /configuration/metrics, 392
GET /configuration/path-alias, 393
PUT /configuration/path-alias, 393
GET /configuration/properties, 401
PUT /configuration/properties, 402
POST /configuration/properties, 402
GET /configuration/properties/(key), 401
PUT /configuration/properties/(key), 403
DELETE /configuration/properties/(key), 403
GET /configuration/purging, 398
PUT /configuration/purging, 398
DELETE /configuration/purging, 399

## /conform

POST /conform, 502

## /cost

POST /cost/(path), 737
GET /cost/estimate/(id), 737

## /deletion-lock

GET /deletion-lock, 389
GET /deletion-lock/(lock-id), 389
PUT /deletion-lock/(lock-id), 391
DELETE /deletion-lock/(lock-id), 391

## /document

GET /document, 539
PUT /document, 542
GET /document/(name), 539
PUT /document/(name), 541
DELETE /document/(name), 543
GET /document/(name)/changes, 542

## /export-location

GET /export-location, 404
GET /export-location/(location-name), 404
PUT /export-location/(location-name), 404
DELETE /export-location/(location-name), 404
GET /export-location/(location-name)/script, 404
PUT /export-location/(location-name)/script, 405

## /external-id

GET /external-id, 405
GET /external-id/(namespace-id), 405
PUT /external-id/(namespace-id), 406
DELETE /external-id/(namespace-id), 406
DELETE /external-id/id/(external-id), 408

## /group

GET /group, 409
PUT /group, 411
DELETE /group, 410
GET /group/(group-name), 409
PUT /group/(group-name), 410
DELETE /group/(group-name), 410
GET /group/(group-name)/children, 413
GET /group/(group-name)/description, 412
PUT /group/(group-name)/description, 412
PUT /group/(group-name)/group/(child-groupname), 413
DELETE /group/(group-name)/group/(child-groupname), 413

GET /group/(group-name)/parents, 412
GET /group/(group-name)/role, 410
PUT /group/(group-name)/user/(username), 413
DELETE /group/(group-name)/user/(username), 414
GET /group/(group-name)/users, 413

## /import

POST /import, 414
GET /import/access, 349
PUT /import/access/group/(group-name), 349
DELETE /import/access/group/(group-name), 349
POST /import/imp, 421
POST /import/placeholder, 423
POST /import/placeholder/(item-id), 427
POST /import/placeholder/(item-id)/(component-type), 424
POST /import/placeholder/(item-id)/(component-type)/adopt/(file-id), 429
POST /import/placeholder/(item-id)/(component-type)/raw, 425
POST /import/placeholder/(item-id)/raw-passkey, 428
POST /import/project, 642
POST /import/project/sequence, 641
POST /import/raw, 416
POST /import/raw-passkey, 418
GET /import/settings, 431
POST /import/settings, 430
GET /import/settings/(settings-id), 431
PUT /import/settings/(settings-id), 432
DELETE /import/settings/(settings-id), 432
POST /import/sidecar/(item-id), 429
POST /import/sidecar/(item-id)/raw, 430
GET /import/storage, 423

## /item

GET /item, 439
PUT /item, 446
DELETE /item, 446
GET /item/(id)/metadata/export/scc, 593
GET /item/(id)/metadata/export/ttml, 594
GET /item/(id)/relation, 459
DELETE /item/(id)/relation, 462
GET /item/(id)/sequence, 464
GET /item/(id)/sequence/(format), 464
PUT /item/(id)/sequence/(format), 464
DELETE /item/(id)/sequence/(format), 464
POST /item/(id)/sequence/conform-metadata, 465
GET /item/(id)/sequence/export, 646
POST /item/(id)/sequence/export, 644
POST /item/(id)/sequence/render, 465
GET /item/(id)/shape, 466
POST /item/(id)/shape, 469
DELETE /item/(id)/shape/, 469
GET /item/(id)/shape/(shape-id), 467
DELETE /item/(id)/shape/(shape-id), 468
GET /item/(id)/shape/(shape-id)/component, 492
POST /item/(id)/shape/(shape-id)/component, 492
GET /item/(id)/shape/(shape-id)/component/(componer 492
DELETE /item/(id)/shape/(shape-id)/component/(compo 494
POST /item/(id)/shape/(shape-id)/component/(compone 494
POST /item/(id)/shape/(shape-id)/component/(compone 494
PUT /item/(id)/shape/(shape-id)/component/(componer 495
DELETE /item/(id)/shape/(shape-id)/component/(compo 495
POST /item/(id)/shape/(shape-id)/component/(compone 493
POST /item/(id)/shape/(shape-id)/component/(compone 493
POST /item/(id)/shape/(shape-id)/component/placehol 495
GET /item/(id)/shape/(shape-id)/cpl, 468
GET /item/(id)/shape/(shape-id)/file, 476
GET /item/(id)/shape/(shape-id)/graph, 468
GET /item/(id)/shape/(shape-id)/graph/dot, 468
GET /item/(id)/shape/(shape-id)/mime/, 478
PUT /item/(id)/shape/(shape-id)/mime/(mime-type), 478
DELETE /item/(id)/shape/(shape-id)/mime/(mime-type) 478
PUT /item/(id)/shape/(shape-id)/placeholder, 475
POST /item/(id)/shape/(shape-id)/version, 475
PUT /item/(id)/shape/(shape-id)/version/(new-versio 475
POST /item/(id)/shape/create, 470
POST /item/(id)/shape/essence, 473
POST /item/(id)/shape/essence/raw, 474
POST /item/(id)/shape/imp, 471
POST /item/(id)/shape/placeholder, 475

POST /item/(id)/shape/raw, 470
GET /item/(id)/shape/version, 473
GET /item/(id)/shape/version/(version), 474
DELETE /item/(id)/shape/version/(version), 474
GET /item/(id)/timeline, 505
DELETE /item/(id)/timeline, 506
GET /item/(id)/timeline/(timeline-format), 505
PUT /item/(id)/timeline/(timeline-format), 506
DELETE /item/(id)/timeline/(timeline-format), 506
POST /item/(id)/timeline/(timeline-format), 503
POST /item/(id1)/relation/(id2), 460
DELETE /item/(id1)/relation/(id2), 462
GET /item/(item-id), 443
DELETE /item/(item-id), 445
POST /item/(item-id)/analyze, 481
GET /item/(item-id)/collections, 452
POST /item/(item-id)/export, 433
POST /item/(item-id)/export/imp, 434
GET /item/(item-id)/library, 453
GET /item/(item-id)/lock, 457
PUT /item/(item-id)/lock, 458
POST /item/(item-id)/lock, 457
DELETE /item/(item-id)/lock, 458
GET /item/(item-id)/loudness, 482
PUT /item/(item-id)/loudness, 483
GET /item/(item-id)/metadata/bulky/, 533
PUT /item/(item-id)/metadata/bulky/, 532
DELETE /item/(item-id)/metadata/bulky/, 533
GET /item/(item-id)/metadata/bulky/(key), 533
PUT /item/(item-id)/metadata/bulky/(key), 535
DELETE /item/(item-id)/metadata/bulky/(key), 535
GET /item/(item-id)/metadata/bulky/(key)/(s-filen, 534
GET /item/(item-id)/posterresource, 497
PUT /item/(item-id)/posterresource, 498
PUT /item/(item-id)/re-index, 451
POST /item/(item-id)/shape/(shape-id)/analyze, 480
POST /item/(item-id)/shape/(shape-id)/export, 436
POST /item/(item-id)/shape/(shape-id)/export/imp, 438
GET /item/(item-id)/shape/(shape-id)/filename, 709
PUT /item/(item-id)/shape/(shape-id)/filename/(stor 709
DELETE /item/(item-id)/shape/(shape-id)/filename/(s 710
POST /item/(item-id)/shape/(shape-id)/highlight-rer 489
GET /item/(item-id)/shape/(shape-id)/highlighter-ec 489
GET /item/(item-id)/shape/(shape-id)/metadata/bulky 533
PUT /item/(item-id)/shape/(shape-id)/metadata/bulky 532
DELETE /item/(item-id)/shape/(shape-id)/metadata/bu 533
GET /item/(item-id)/shape/(shape-id)/metadata/bulky 533
PUT /item/(item-id)/shape/(shape-id)/metadata/bulky 535
DELETE /item/(item-id)/shape/(shape-id)/metadata/bu 535
GET /item/(item-id)/shape/(shape-id)/metadata/bulky 534
GET /item/(item-id)/shape/(shape-id)/smartcrop-edl, 491
POST /item/(item-id)/shape/(shape-id)/smartcrop-rer 491
GET /item/(item-id)/shape/(shape-id)/storage-rule, 711
GET /item/(item-id)/shape/(shape-id)/tag/, 477
PUT /item/(item-id)/shape/(shape-id)/tag/(tag-name) 477
DELETE /item/(item-id)/shape/(shape-id)/tag/(tag-na 477
POST /item/(item-id)/shape/(shape-id)/thumbnail, 472
POST /item/(item-id)/shape/(shape-id)/transcode, 501
POST /item/(item-id)/shape/(shape-id)/update, 476
POST /item/(item-id)/thumbnail, 495
GET /item/(item-id)/thumbnail/spritesheet, 498
GET /item/(item-id)/thumbnailresource, 497
PUT /item/(item-id)/thumbnailresource, 497
POST /item/(item-id)/transcode, 500
GET /item/(item-id)/uri, 456
GET /item/(item-id)/waveform/alltracks, 487
GET /item/(item-id)/waveform/data, 484
GET /item/(item-id)/waveform/image, 485
GET /item/(item-id)/waveform/imageURI,

486
GET /item/(item-id)/waveform/values, 483
POST /item/access, 347
DELETE /item/access, 347
GET /item/history, 451
POST /item/list, 451

## /javascript

GET /javascript/session, 507
GET /javascript/session/(id), 507
DELETE /javascript/session/(id), 508
POST /javascript/test, 507

## /job

GET /job, 508
POST /job, 514
DELETE /job, 515
GET /job/(job-id), 510
PUT /job/(job-id), 513
DELETE /job/(job-id), 513
GET /job/(job-id)/problem, 515
POST /job/(job-id)/re-run, 513
GET /job/problem, 514
PUT /job/search, 510

## /jobtype

GET /jobtype, 517

## /library

GET /library, 518
POST /library, 518
DELETE /library, 519
GET /library/(library-id), 521
PUT /library/(library-id), 524
DELETE /library/(library-id), 519
PUT /library/(library-id)/(item-id), 525
DELETE /library/(library-id)/(item-id), 525
POST /library/(library-id)/export, 436
PUT /library/(library-id)/item-metadata, 525
POST /library/(library-id)/list, 526
PUT /library/(library-id)/re-index, 527
GET /library/(library-id)/settings, 520
PUT /library/(library-id)/settings, 521
PUT /library/re-index, 527

## /license

GET /license, 528
GET /license/slave, 529
PUT /license/slave, 528
POST /license/slave, 528
GET /license/slave/(id), 529
DELETE /license/slave/(id), 529
GET /license/slave/(id)/license, 529
GET /license/slave/license, 529
GET /license/slave/license/(id), 529

## /log

GET /log, 354
GET /log/export, 356
GET /log/transfer-log, 721

## /metadata

GET /metadata, 536
PUT /metadata, 537
GET /metadata/(uuid), 538
DELETE /metadata/(uuid), 538
GET /metadata/dataset, 584
GET /metadata/dataset/(name), 585
PUT /metadata/dataset/(name), 586
DELETE /metadata/dataset/(name), 587
GET /metadata/migration, 588
POST /metadata/migration, 588
GET /metadata/migration/(id), 588

## /metadata-field

GET /metadata-field, 570
GET /metadata-field/(field-name), 571
PUT /metadata-field/(field-name), 571
DELETE /metadata-field/(field-name), 571
GET /metadata-field/(field-name)/allowed-values, 572
POST /metadata-field/(field-name)/allowed-values, 572
GET /metadata-field/(field-name)/merged-access/, 575
GET /metadata-field/(field-name)/values, 573
PUT /metadata-field/(field-name)/values, 574
GET /metadata-field/field-group, 579
PUT /metadata-field/field-group, 582
GET /metadata-field/field-group/(group-name), 581
PUT /metadata-field/field-group/(group-name), 580
DELETE /metadata-field/field-group/(group-name), 581
PUT /metadata-field/field-group/(group-name)/(field 581
DELETE /metadata-field/field-group/(group-name)/(fi 582
GET /metadata-field/field-group/(group-name)/merged 577
PUT /metadata-field/field-group/(parent-group-name) 582

DELETE /metadata-field/field-group/(parent-group-name)/(child-group-name), 582
GET /metadata-field/field-group/merged-access/, 576
GET /metadata-field/merged-access/, 575
GET /metadata-field/terse-schema, 574

## /metadata-schema

GET /metadata-schema, 590
PUT /metadata-schema, 591
DELETE /metadata-schema, 592
GET /metadata-schema/(group-name), 592
PUT /metadata-schema/(group-name), 592
DELETE /metadata-schema/(group-name), 593

## /projection

GET /projection, 589
DELETE /projection/(projection-id), 590
GET /projection/(projection-id)/incoming, 589
PUT /projection/(projection-id)/incoming, 590
GET /projection/(projection-id)/outgoing, 589
PUT /projection/(projection-id)/outgoing, 589

## /quota

GET /quota/, 647
POST /quota/, 646
GET /quota/(rule-id), 648
DELETE /quota/(rule-id), 648
PUT /quota/(rule-id)/evaluate, 648

## /reindex

GET /reindex/(index), 568
PUT /reindex/(index), 568

## /relation

POST /relation, 460
GET /relation/(relation-id), 461
PUT /relation/(relation-id), 461
DELETE /relation/(relation-id), 462

## /resource

GET /resource, 649
POST /resource, 649
GET /resource/(type), 649
POST /resource/(type), 649
GET /resource/(type)/(resource-id), 650
PUT /resource/(type)/(resource-id), 649
DELETE /resource/(type)/(resource-id), 650
PUT /resource/(type)/(resource-id)/configuration, 651
GET /resource/(type)/(resource-id)/configuration/pr
650
GET /resource/(type)/(resource-id)/status, 650
POST /resource/(type)/(resource-id)/sync, 246

## /scheduled-request

GET /scheduled-request, 652
DELETE /scheduled-request/, 653
GET /scheduled-request/(request-id), 652
DELETE /scheduled-request/(request-id), 653
GET /scheduled-request/(request-id)/response, 653

## /search

GET /search, 654
PUT /search, 657
PUT /search/autocomplete, 663
GET /search/file, 662
PUT /search/file, 662
POST /search/optimize, 665
GET /search/shape, 661
PUT /search/shape, 661

## /selftest

GET /selftest, 666
GET /selftest/(test-name), 666

## /sequence

POST /sequence/export, 644
POST /sequence/render, 465

## /shape-tag

GET /shape-tag/, 667
GET /shape-tag/(tag-name), 667
PUT /shape-tag/(tag-name), 667
DELETE /shape-tag/(tag-name), 668
GET /shape-tag/(tag-name)/item/(item-id)/shape/(sha
668
GET /shape-tag/(tag-name)/script, 668
PUT /shape-tag/(tag-name)/script, 668
DELETE /shape-tag/(tag-name)/script, 668

## /site

GET /site, 669
GET /site/(site-id), 669
PUT /site/(site-id), 669

## /storage

GET /storage, 695

POST /storage, 697
GET /storage/(storage-id), 698
PUT /storage/(storage-id), 698
DELETE /storage/(storage-id), 699
GET /storage/(storage-id)/auto-import/, 673
PUT /storage/(storage-id)/auto-import/, 672
DELETE /storage/(storage-id)/auto-import/, 674
PUT /storage/(storage-id)/auto-import/disable, 673
PUT /storage/(storage-id)/auto-import/enable, 673
PUT /storage/(storage-id)/evacuate, 704
DELETE /storage/(storage-id)/evacuate, 704
GET /storage/(storage-id)/export, 703
GET /storage/(storage-id)/freespace, 700
GET /storage/(storage-id)/method, 700
PUT /storage/(storage-id)/method, 701
DELETE /storage/(storage-id)/method, 703
GET /storage/(storage-id)/method/(method-id), 702
PUT /storage/(storage-id)/method/(method-id), 702
DELETE /storage/(storage-id)/method/(method-id), 703
POST /storage/(storage-id)/rescan, 700
GET /storage/(storage-id)/status, 700
PUT /storage/(storage-id)/type/(type), 700
GET /storage/auto-import/, 673
POST /storage/import, 703
GET /storage/storage-group/, 708
GET /storage/storage-group/(group-name), 708
PUT /storage/storage-group/(group-name), 708
DELETE /storage/storage-group/(group-name), 708
PUT /storage/storage-group/(group-name)/(storage-id), 709
DELETE /storage/storage-group/(group-name)/(storage-id), 709

## /storage-rule

GET /storage-rule/, 710

## /task-definition

GET /task-definition, 714
POST /task-definition, 714
GET /task-definition/(task-id), 715
PUT /task-definition/(task-id), 715
DELETE /task-definition/(task-id), 715
GET /task-definition/(task-id)/script, 716
PUT /task-definition/(task-id)/script, 716
GET /task-definition/(task-id)/validate, 715
GET /task-definition/jobtype/(type), 714
POST /task-definition/jobtype/(type), 716
DELETE /task-definition/jobtype/(type), 716
GET /task-definition/jobtype/(type)/graph, 717
GET /task-definition/jobtype/(type)/graph/dot, 717
GET /task-definition/jobtype/(type)/step/(step), 715
PUT /task-definition/jobtype/(type)/step/(step), 715
DELETE /task-definition/jobtype/(type)/step/(step), 715
GET /task-definition/jobtype/(type)/step/(step)/scr 716
PUT /task-definition/jobtype/(type)/step/(step)/scr 716
GET /task-definition/jobtype/(type)/step/(step)/val 715

## /task-group

GET /task-group, 717
DELETE /task-group, 719
GET /task-group/(group-name), 719
PUT /task-group/(group-name), 718
DELETE /task-group/(group-name), 719
PUT /task-group/(group-name)/transcoder/(transcoder 719
DELETE /task-group/(group-name)/transcoder/(transco 719

## /token

GET /token, 735

## /transfer

GET /transfer, 720
GET /transfer/(transfer-id), 720
PUT /transfer/(transfer-id), 721

## /user

GET /user, 722
PUT /user, 725
POST /user, 722
GET /user/(username), 723
PUT /user/(username), 723

DELETE /user/(username), 724
GET /user/(username)/access, 726
PUT /user/(username)/alias/(name), 734
DELETE /user/(username)/alias/(name), 734
GET /user/(username)/allgroups, 729
PUT /user/(username)/enable, 725
GET /user/(username)/groups, 728
PUT /user/(username)/groups, 729
GET /user/(username)/key, 732
POST /user/(username)/key, 732
GET /user/(username)/key/(key-id), 732
PUT /user/(username)/key/(key-id), 733
DELETE /user/(username)/key/(key-id), 733
PUT /user/(username)/password, 727
GET /user/(username)/password/salt, 728
POST /user/(username)/password/salt, 728
GET /user/(username)/realname, 726
PUT /user/(username)/realname, 727
GET /user/(username)/roles, 729
GET /user/(username)/token, 736
PUT /user/(username)/validate, 727
GET /user/graph, 731
GET /user/graph/dot, 731

## /version

GET /version, 528

## /vidispine-logs

GET /vidispine-logs, 738
GET /vidispine-logs/job, 739
POST /vidispine-logs/job, 739
GET /vidispine-logs/job/(job-id), 739
DELETE /vidispine-logs/job/(job-id), 739
POST /vidispine-logs/job/(job-id)/upload, 740
DELETE /vidispine-logs/job/(job-id)/upload, 740
GET /vidispine-logs/job/(job-id)/uri, 740

## /vidispine-service

GET /vidispine-service, 740
PUT /vidispine-service/disable, 741
PUT /vidispine-service/enable, 741
GET /vidispine-service/service/(service), 741
PUT /vidispine-service/service/(service), 741
PUT /vidispine-service/service/(service)/disable, 741
GET /vidispine-service/stacktrace, 741

## /vxa

GET /vxa, 742
GET /vxa/(uuid), 742
DELETE /vxa/(uuid), 743
GET /vxa/(uuid)/configuration, 742
PUT /vxa/enable-ssh, 742

## /whoami

GET /whoami, 599

## /{access-entity}

GET {access-entity}/(entity-id)/access, 345
POST {access-entity}/(entity-id)/access, 345
GET {access-entity}/(entity-id)/access/(access-id), 346
DELETE {access-entity}/(entity-id)/access/(access-i 346
POST {access-entity}/(entity-id)/access/bulk, 347
DELETE {access-entity}/(entity-id)/access/bulk, 348
GET {access-entity}/(entity-id)/access/graph, 353
GET {access-entity}/(entity-id)/access/graph/dot, 353
PUT {access-entity}/(entity-id)/access/owner/(usern 347
GET {access-entity}/(entity-id)/merged-access/, 350
GET {access-entity}/(entity-id)/merged-access/group 352

## /{eidr-content-resource}

PUT {eidr-content-resource}/eidr/sync, 325

## /{external-id-resource}

GET {external-id-resource}, 407
DELETE {external-id-resource}, 408
PUT {external-id-resource}/(external-id), 407
DELETE {external-id-resource}/(external-id), 408

## /{field-access-resource}

GET {field-access-resource}, 577
POST {field-access-resource}, 578
DELETE {field-access-resource}/(access-id), 579

## /{file-resource}

GET {file-resource}, 678

DELETE {file-resource}, 691
PUT {file-resource}/abandon, 693
POST {file-resource}/analyze, 694
POST {file-resource}/analyze/imp, 694
GET {file-resource}/data, 680
POST {file-resource}/data, 680
DELETE {file-resource}/entity, 692
PUT {file-resource}/hash/(hash), 693
POST {file-resource}/import, 684
POST {file-resource}/import/assetmap, 686
POST {file-resource}/path, 692
PUT {file-resource}/re-index, 693
PUT {file-resource}/restore, 301
GET {file-resource}/shape, 695
PUT {file-resource}/state/(state), 692
POST {file-resource}/storage/(target-storage-id), 690
POST {file-resource}/uri, 680

## /{item-content-resource}

GET {item-content-resource}, 453

## /{key-value-metadata-resource}

GET {key-value-metadata-resource}, 544
PUT {key-value-metadata-resource}, 545
DELETE {key-value-metadata-resource}, 545
GET {key-value-metadata-resource}/(keypath), 546
PUT {key-value-metadata-resource}/(keypath), 546
DELETE {key-value-metadata-resource}/(keypath), 546
PUT {key-value-metadata-resource}/(prefix), 545

## /{lock-entity}

GET {lock-entity}/deletion-lock, 390
POST {lock-entity}/deletion-lock, 390
GET {lock-entity}/deletion-lock/(lock-id), 390
PUT {lock-entity}/deletion-lock/(lock-id), 391
DELETE {lock-entity}/deletion-lock/(lock-id), 391

## /{metadata-entity}

GET {metadata-entity}/(entity-id)/metadata, 547
PUT {metadata-entity}/(entity-id)/metadata, 550
GET {metadata-entity}/(entity-id)/metadata/changes, 552
PUT {metadata-entity}/(entity-id)/metadata/changes, 561
GET {metadata-entity}/(entity-id)/metadata/changes, 555
PUT {metadata-entity}/(entity-id)/metadata/changes, 558
DELETE {metadata-entity}/(entity-id)/metadata/chang 563
GET {metadata-entity}/(entity-id)/metadata/changes, 555
PUT {metadata-entity}/(entity-id)/metadata/changes, 561
PUT {metadata-entity}/(entity-id)/metadata/changes, 561
PUT {metadata-entity}/(entity-id)/metadata/entry, 566
PUT {metadata-entity}/(entity-id)/metadata/entry/(e 565
GET {metadata-entity}/(entity-id)/metadata/graph, 567
GET {metadata-entity}/(entity-id)/metadata/graph/do 567
PUT {metadata-entity}/(entity-id)/metadata/move, 551

## /{metadata-resource}

GET {metadata-resource}/(id)/metadata-lock, 569
POST {metadata-resource}/(id)/metadata-lock, 569
GET {metadata-resource}/(id)/metadata-lock/(lock-id 569
PUT {metadata-resource}/(id)/metadata-lock/(lock-id 570
DELETE {metadata-resource}/(id)/metadata-lock/(lock 570

## /{notification-entity}

GET {notification-entity}/(entity-id)/notification, 633
POST {notification-entity}/(entity-id)/notificatio 634
DELETE {notification-entity}/(entity-id)/notificati 634
GET {notification-entity}/(entity-id)/notification/ 634
PUT {notification-entity}/(entity-id)/notification/ 635
DELETE {notification-entity}/(entity-id)/notificati 634

## /{notification-resource}

GET {notification-resource}/, 631
POST {notification-resource}/, 631

DELETE {notification-resource}/, 632
GET {notification-resource}/(notification-id),
632
PUT {notification-resource}/(notification-id),
634
DELETE {notification-resource}/(notification-id),
633

## /{rule-resource}

GET {rule-resource}, 711
PUT {rule-resource}, 712
DELETE {rule-resource}, 712
GET {rule-resource}/(tag-name), 713
PUT {rule-resource}/(tag-name), 713
DELETE {rule-resource}/(tag-name), 714

## /{site-rule-entity}

GET {site-rule-entity}/site-rule, 670
POST {site-rule-entity}/site-rule, 671
GET {site-rule-entity}/site-rule/(id),
671
PUT {site-rule-entity}/site-rule/(id),
672
DELETE {site-rule-entity}/site-rule/(id),
672

## /{storage-resource}

GET {storage-resource}/file, 674
POST {storage-resource}/file, 691
POST {storage-resource}/file/data, 679
POST {storage-resource}/file/import, 687
POST {storage-resource}/file/import/assetmap,
688
GET {storage-resource}/importable, 682

## /{thumbnail-resource}

GET {thumbnail-resource}, 498
DELETE {thumbnail-resource}, 499
GET {thumbnail-resource}/(time), 499
PUT {thumbnail-resource}/(time), 499
DELETE {thumbnail-resource}/(time), 500
POST {thumbnail-resource}/(time)/export,
500

## A

activeMQAdminUrl
configuration property, 280
activeMQBrokerName
configuration property, 281
additionalHash
reserved key, 704
alwaysGenerateThumbnails
configuration property, 275
api.dataType() (api method), 291
api.delete() (api method), 291
api.get() (api method), 291
api.getInfo() (api method), 292
api.header() (api method), 291
api.input() (api method), 291
api.path() (api method), 290
api.post() (api method), 291
api.put() (api method), 291
api.queryParam() (api method), 290
api.rich() (api method), 292
api.timeout() (api method), 291
api.user() (api method), 291
apiNoauthUri
configuration property, 270
apiUri
configuration property, 270
auditTrailIncludeBody
configuration property, 281
auditTrailPurgingBatch
configuration property, 281
auditTrailPurgingCompress
configuration property, 281
auditTrailPurgingDirectory
configuration property, 281
auditTrailPurgingTime
configuration property, 281
autoRemoveExpiredDeletionLocks
configuration property, 283
azureSasValidTime
configuration property, 277

## B

bulkyMetadataKeysToIgnore
configuration property, 282
bulkyMetadataMigrationThreads
configuration property, 273

## C

cerifyPriority
job metadata key, 517
changeLogForcePurgingTime
configuration property, 281
changeLogPurgingTime
configuration property, 281
checksumMode
job metadata key, 517
closeLimit
reserved key, 704
clusterName
configuration property, 270
com.vidispine.credentials.dir, 149, 300
com.vidispine.site, 6, 937
compressDocumentMessages
configuration property, 280
concurrentJobs
configuration property, 274
configuration property
activeMQAdminUrl, 280
activeMQBrokerName, 281
alwaysGenerateThumbnails, 275
apiNoauthUri, 270
apiUri, 270
auditTrailIncludeBody, 281
auditTrailPurgingBatch, 281
auditTrailPurgingCompress, 281
auditTrailPurgingDirectory, 281
auditTrailPurgingTime, 281
autoRemoveExpiredDeletionLocks, 283
azureSasValidTime, 277
bulkyMetadataKeysToIgnore, 282
bulkyMetadataMigrationThreads, 273
changeLogForcePurgingTime, 281
changeLogPurgingTime, 281

clusterName, 270
compressDocumentMessages, 280
concurrentJobs, 274
dedicatedJobPool, 274
defaultIngestStorage, 274
defaultStorageRuleJobPriority, 277
defaultTranscoder, 271
deletionLockCleanUpBatchSzie, 283
disableATime, 279
disableMetadataSchema, 273
disableSequenceChecker, 282
disableSiteCrunching, 270
disableThumbnailGeneration, 275
disableThumbnailReindexing, 275
elasticsearchBulkBuffer, 272
elasticsearchPath, 271
elasticsearchWorkerCount, 272
enableTranscoderHash, 276
enableTranscoderTransfer, 279
fileGrowingTimeout, 280
fileHashAlgorithm, 276
fileHashExecutionTime, 280
fileHierarchy, 278
fileNotGrowingTimeout, 280
fileSequenceStart, 278
fileTempKeyDuration, 276
firstLastModifiedAsCreationTime, 279
glacierArchiveDescription, 278
groupImportableFiles, 275
indexCollectionItemOrder, 272
indexDocumentMetadata, 273
indexFieldGroups, 272
indexQueueLimit, 273
indexTimespans, 272
itemDeleteExecutionTime, 280
itemDeleteInterval, 280
itemDeleteIntervalShort, 280
javascriptInterpreter, 280
jobExclusiveStepMaxWait, 274
jobPurgingDirectory, 282
jobPurgingTime, 281
jobRetryCount, 274
keepEmptyDirectories, 275
keepMissingFiles, 275
ldapAuthentication, 274
legacyTransientFieldTypes, 273
libraryExpireTime, 279
libraryUpdateInterval, 279
localFSTimeData, 279
maxFileMetadataLength, 274
maxSearchResults, 273
maxTranscoderUnavailableTime, 282
mediaCheckInterval, 275
parseFileMetadata, 274
parseXMP, 275
passwordHashAlgorithm, 274
s3ConcurrentParts, 276
s3ConnectionTimeout, 276
s3CredentialType, 277
s3MaxErrorRetry, 277
s3PartSize, 276
s3PartSizeIncrease, 276
s3ProxyValidTime, 276
s3SocketTimeout, 276
scanMethodAlgorithm, 275
signiantManagerHost, 279
signiantManagerPassword, 279
signiantManagerUser, 279
simpleImageProcessor, 275
skipLibraryIndexUpdates, 273
slaveLicenseProxy, 270
solrAutoSoftCommit, 272
solrCollection, 271
solrCommitInterval, 272
solrDeleteMergeSize, 272
solrGroupLimit, 271
solrPath, 271
solrPingAttempts, 271
solrPingTimeout, 271
solrQueryTimeout, 271
solrSoftCommitInterval, 272
solrUpdateQueueSize, 272
statsPerSecond, 279
storageActivationFile, 276
storageRuleDisableArchiveSources, 279
stornextFileMetadata, 277
stsAssumeRole, 277
stsCredentialDuration, 277
stsRegion, 277
syncVxaDeletes, 282
syncVxaFileChanges, 282
thumbnailHierarchy, 278
transcoderNonblockingStatusInterval, 282
trustArchivedFiles, 278
useAbsoluteSccTimeCode, 273
useAzureProxy, 277
useLucene, 279
useMutableRangeWrites, 277
userTokenDefaultInterval, 274
userTokenMaxInterval, 274
userTokenRefreshInterval, 274
useS3Proxy, 276
useSegmentFiles, 277
useVxaHash, 282
useVxaMimeType, 282
validatexml, 270
xmpIgnoreElements, 275
zkHost, 271

constants
storage states, 139
storage types, 139
context.getChannel() (context method), 161
context.getComponent() (context method), 160
context.getExtension() (context method), 161
context.getFileId() (context method), 161
context.getItem() (context method), 160
context.getJobId() (context method), 161
context.getJobMetadata() (context method), 160
context.getJobType() (context method), 161
context.getOriginalComponentFilename() (context method), 161
context.getOriginalFilename() (context method), 161
context.getShape() (context method), 160
context.getStorage() (context method), 160
context.getTags() (context method), 161

## D

dedicatedJobPool
configuration property, 274
defaultIngestStorage
configuration property, 274
defaultStorageRuleJobPriority
configuration property, 277
defaultTranscoder
configuration property, 271
deleteFileIfNotFound
reserved key, 705
deleteFileIfReadOnly
reserved key, 705
deletionLockCleanUpBatchSzie
configuration property, 283
detectRenamedFiles
reserved key, 705
disableATime
configuration property, 279
disabledSidecarExtensions
reserved key, 155
disableMetadataSchema
configuration property, 273
disableSequenceChecker
configuration property, 282
disableSiteCrunching
configuration property, 270
disableThumbnailGeneration
configuration property, 275
disableThumbnailReindexing
configuration property, 275

## E

elasticsearchBulkBuffer
configuration property, 272
elasticsearchPath
configuration property, 271
elasticsearchWorkerCount
configuration property, 272
enableTranscoderHash
configuration property, 276
enableTranscoderTransfer
configuration property, 279
environment variable
com.vidispine.asyncpool.coresize, 284
com.vidispine.credentials.dir, 149, 283, 300
com.vidispine.license.dir, 283
com.vidispine.license.tmpdir, 283
com.vidispine.log.dir, 283
com.vidispine.service.quorum, 284
com.vidispine.site, 6, 283, 937
com.vidispine.xml.prefix, 283
infi.sleep_after_start, 283
vidispine.identifier.format, 6, 283, 937
excludeFilter
reserved key, 705

## F

file.getAllMetadata() (file method), 299
file.getMetadata() (file method), 298
file.setMetadata() (file method), 298
fileGrowingTimeout
configuration property, 280
fileHashAlgorithm
configuration property, 276
fileHashExecutionTime
configuration property, 280
fileHierarchy
configuration property, 278
fileListBatchSize
reserved key, 704
fileNotGrowingTimeout
configuration property, 280
fileSequenceStart
configuration property, 278
fileTempKeyDuration
configuration property, 276
firstLastModifiedAsCreationTime
configuration property, 279

## G

glacierArchiveDescription
configuration property, 278
groupImportableFiles
configuration property, 275

## H

hashingThreadCount
reserved key, 706
hashMode

reserved key, 704
http.followRedirects() (http method), 292
http.proxy() (http method), 293
http.uri() (http method), 292

## I

ignoreSidecarImport
reserved key, 155
indexCollectionItemOrder
configuration property, 272
indexDocumentMetadata
configuration property, 273
indexFieldGroups
configuration property, 272
indexQueueLimit
configuration property, 273
indexTimespans
configuration property, 272
itemDeleteExecutionTime
configuration property, 280
itemDeleteInterval
configuration property, 280
itemDeleteIntervalShort
configuration property, 280

## J

javascriptInterpreter
configuration property, 280
job metadata key
cerifyPriority, 517
checksumMode, 517
lastSmpteTimeCode, 517
smpteTimeCode, 517
job.containsKey() (job method), 173
job.deleteData() (job method), 172
job.fail() (job method), 172
job.fatalFail() (job method), 173
job.getData() (job method), 172
job.getDataOrDefault() (job method), 173
job.getId() (job method), 172
job.getKeys() (job method), 173
job.getStepId() (job method), 173
job.getUser() (job method), 173
job.log() (job method), 172
job.setData() (job method), 172
job.vidinetCost() (job method), 175
job.vidinetJob() (job method), 175
job.wait() (job method), 173
job.waitForJobs() (job method), 173
jobExclusiveStepMaxWait
configuration property, 274
jobPurgingDirectory
configuration property, 282
jobPurgingTime
configuration property, 281
jobRetryCount
configuration property, 274

## K

keepEmptyDirectories
configuration property, 275
reserved key, 705
keepMissingFiles
configuration property, 275
reserved key, 704

## L

lastSmpteTimeCode
job metadata key, 517
ldapAuthentication
configuration property, 274
legacyTransientFieldTypes
configuration property, 273
libraryExpireTime
configuration property, 279
libraryUpdateInterval
configuration property, 279
localFSTimeData
configuration property, 279
logger.json() (logger method), 294
logger.log() (logger method), 294
lostLimit
reserved key, 704

## M

maxFileMetadataLength
configuration property, 274
maxSearchResults
configuration property, 273
maxTranscoderUnavailableTime
configuration property, 282
mediaCheckInterval
configuration property, 275
metadatahelper.createMetadata() (metadatahelper method), 294
metadatahelper.createMetadataGroup() (metadatahelper method), 295
metadatahelper.createMetadataTimespan() (metadatahelper method), 294
metadatahelper.generateMetadataField() (metadatahelper method), 295
metadatahelper.log() (metadatahelper method), 295
metadatahelper.metadataToStr() (metadatahelper method), 295

## N

noDefaultHash

reserved key, 706
notification.send() (notification method), 295

## P

parseFileMetadata
configuration property, 274
parseXMP
configuration property, 275
passwordHashAlgorithm
configuration property, 274
probeFileBeforeClosing
reserved key, 705

## R

refreshInterval
reserved key, 704
refreshOnStart
reserved key, 705
reserved key
additionalHash, 704
closeLimit, 704
deleteFileIfNotFound, 705
deleteFileIfReadOnly, 705
detectRenamedFiles, 705
disabledSidecarExtensions, 155
excludeFilter, 705
fileListBatchSize, 704
hashingThreadCount, 706
hashMode, 704
ignoreSidecarImport, 155
keepEmptyDirectories, 705
keepMissingFiles, 704
lostLimit, 704
noDefaultHash, 706
probeFileBeforeClosing, 705
refreshInterval, 704
refreshOnStart, 705
s3EventETag, 706
s3EventTime, 706
scanMethodAlgorithm, 707
scanOnStart, 705
snsTopic, 706
sqsEndpoint, 706
sqsName, 706
statsPerSecond, 705
storageActivationFile, 707
toAppearLimit, 704
verifyHashAfterTransfer, 706
vxaId, 706

## S

s3ConcurrentParts
configuration property, 276
s3ConnectionTimeout
configuration property, 276
s3CredentialType
configuration property, 277
s3EventETag
reserved key, 706
s3EventTime
reserved key, 706
s3MaxErrorRetry
configuration property, 277
s3PartSize
configuration property, 276
s3PartSizeIncrease
configuration property, 276
s3ProxyValidTime
configuration property, 276
s3SocketTimeout
configuration property, 276
scanMethodAlgorithm
configuration property, 275
reserved key, 707
scanOnStart
reserved key, 705
shell.exec() (shell method), 293
signiantManagerHost
configuration property, 279
signiantManagerPassword
configuration property, 279
signiantManagerUser
configuration property, 279
simpleImageProcessor
configuration property, 275
skipLibraryIndexUpdates
configuration property, 273
slaveLicenseProxy
configuration property, 270
smpteTimeCode
job metadata key, 517
snsTopic
reserved key, 706
solrAutoSoftCommit
configuration property, 272
solrCollection
configuration property, 271
solrCommitInterval
configuration property, 272
solrDeleteMergeSize
configuration property, 272
solrGroupLimit
configuration property, 271
solrPath
configuration property, 271
solrPingAttempts
configuration property, 271
solrPingTimeout

configuration property, 271
solrQueryTimeout
configuration property, 271
solrSoftCommitInterval
configuration property, 272
solrUpdateQueueSize
configuration property, 272
sqsEndpoint
reserved key, 706
sqsName
reserved key, 706
statsPerSecond
configuration property, 279
reserved key, 705
storage states
constants, 139
storage types
constants, 139
storageActivationFile
configuration property, 276
reserved key, 707
storageRuleDisableArchiveSources
configuration property, 279
stornextFileMetadata
configuration property, 277
stsAssumeRole
configuration property, 277
stsCredentialDuration
configuration property, 277
stsRegion
configuration property, 277
syncVxaDeletes
configuration property, 282
syncVxaFileChanges
configuration property, 282

## T

thumbnailHierarchy
configuration property, 278
toAppearLimit
reserved key, 704
transcoderNonblockingStatusInterval
configuration property, 282
trustArchivedFiles
configuration property, 278

## U

useAbsoluteSccTimeCode
configuration property, 273
useAzureProxy
configuration property, 277
useLucene
configuration property, 279
useMutableRangeWrites
configuration property, 277
userTokenDefaultInterval
configuration property, 274
userTokenMaxInterval
configuration property, 274
userTokenRefreshInterval
configuration property, 274
useS3Proxy
configuration property, 276
useSegmentFiles
configuration property, 277
useVxaHash
configuration property, 282
useVxaMimeType
configuration property, 282

## V

validatexml
configuration property, 270
verifyHashAfterTransfer
reserved key, 706
vidispine.identifier.format, 6, 937
vxaId
reserved key, 706

## W

wrapper.getBulkyMetadata() (wrapper method), 63
wrapper.getMetadata() (wrapper method), 62
wrapper.getOldBulkyMetadata() (wrapper method), 63
wrapper.getOldMetadata() (wrapper method), 62
wrapper.getShape() (wrapper method), 63
wrapper.getShapeMetadata() (wrapper method), 62
wrapper.setMetadata() (wrapper method), 63

## X

xmpIgnoreElements
configuration property, 275

## Z

zkHost
configuration property, 271