Vidispine REST API Documentation

Release 4.7.2

Vidispine AB

Apr 03, 2017

CONTENTS

1 Introduction and data model 3

2 Items and Metadata 15

3 Collections and Libraries 85

4 Shapes, Components and Transcoding 95

5 Storages and Files 121

6 Jobs and Task Definitions 145

7 Notifications 155

8 Resources 157

9 Timelines and sequences 171

10 Users, Groups, and Access control 183

11 Multi-site 199

12 Monitoring 201

13 Configuration and Integration 207

14 Troubleshooting and obtaining information 253

15 Standalone Vidispine 257

16 API Reference 269

17 Release Notes 751

HTTP Routing Table 799

Index 807

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.

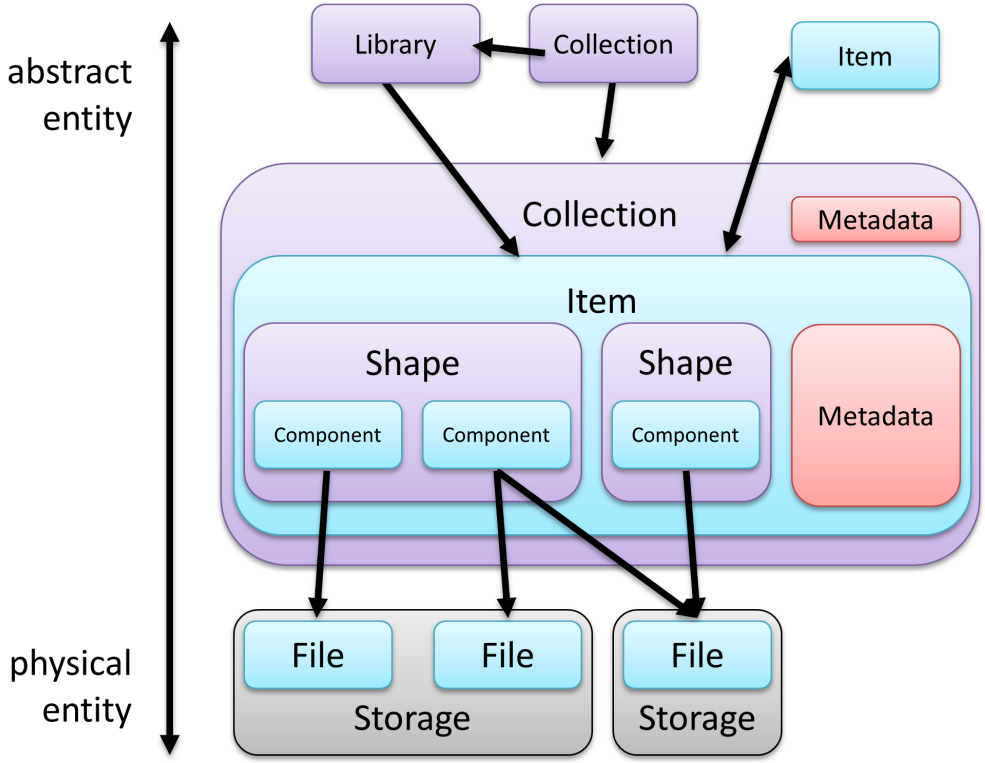


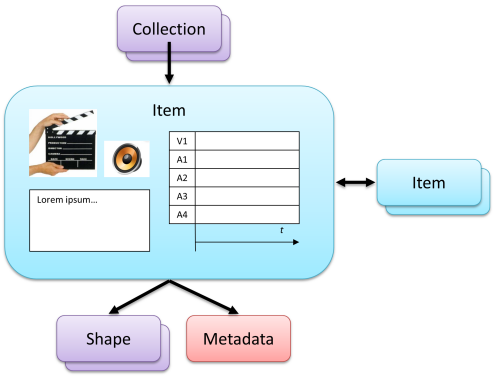
Fig. 1.1: Overview of the entities in Vidispine

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 informa-

tion 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*.

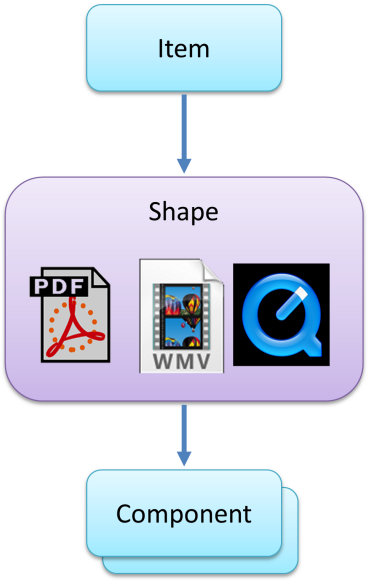
1.1.2 Shape

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

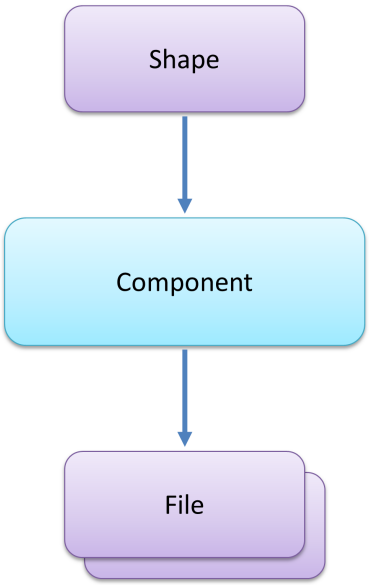
Each component corresponds to one file content. There may be several copies of the file however.

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, *matrix* 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.

Matrix parameters

Matrix parameters are be applied to the *segments* of the path of the URI. The are used to discriminate or modify the result. Before each matrix parameter key/value pair there should be a semicolon (`;`). Equal signs are used to separate key and value, and they have to be URL encoded.

In Vidispine, most of the times, the matrix parameters should be applied to the last segment, but before any query parameters.

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 four main categories related to time which will be discussed here: time bases, time positions (a.k.a. time codes), time intervals and time durations.

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 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 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

New in version 4.5.

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 }*
```

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`

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.

Changed in version 4.6: Updated to use the video timestamp if the SMPTE timecode is missing.

Checksum on file transfer

New in version 4.2.8.

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

New in version 4.5.

Added as BETA in 4.5

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.

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. (New in 4.1.)
`timecode`	A *time code* value. (New in 4.1.)

Changed in version 4.7: The `long` datatype was added.

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. (New in 4.1.)
`timecode-noindex`	A *time code* value. No indexing will take place. (New in 4.1.)

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.

```
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.

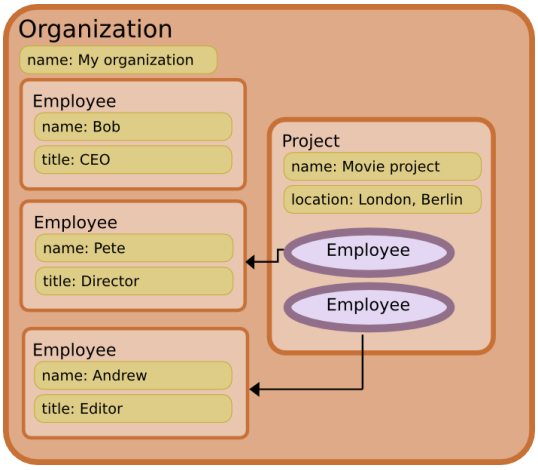
	Group	Nested groups	Field
min	The minimum number of times that the group can occur at top-level.	The minimum number of times that the group can occur inside the given group	The minimum number of times the field can occur inside the given group
max	The maximum number of times that the group can occur 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>
  <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 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>user</name>
          <value change="VX-30" timestamp="2010-03-19T09:08:09.588+01:00" user="system
↪">admin</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>user</name>
               <value change="VX-30" timestamp="2010-03-19T09:08:09.588+01:00" user=
↪"system">admin</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>
```

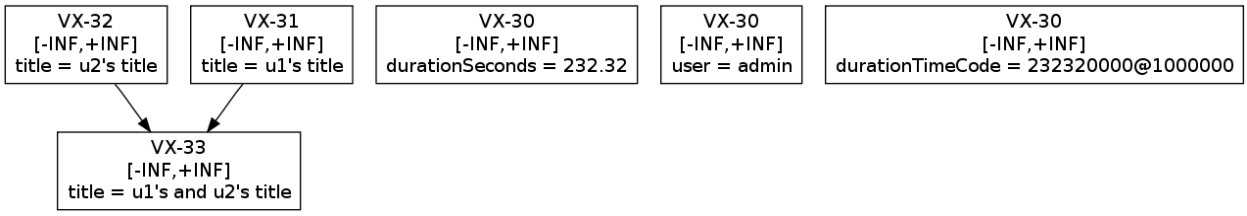
Which results in:

```
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>user</name>
               <value change="VX-30" timestamp="2010-03-19T09:08:09.588+01:00" user=
↪"system">admin</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.



2.3.6 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>
```

2.3.7 Metadata defined by the systems

Name	Description
user	The name of the user that imported 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.
shapeTag	A shape tag that is used on a shape belonging to the item.
created	The time when item was created.
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.

Changed in version 4.6: The PDF and document media types were added.

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
__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_size	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
__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

For collections, the following transient metadata fields exist:

Name	Description	Indexed
__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. (New in 4.6.)	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

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

Item and collection searching in Vidispine is implemented using Solr as 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`).

Search intervals

By setting <*interals*> 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

New in version 4.3.

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> <text>bar</text>	foo AND bar	foo OR bar
<field>foo</field> <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.

[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.

	`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

New in version 4.2.3.

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*.

Example

```
<ItemSearchDocument>
   <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

New in version 4.2.2.

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.

[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.

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

New in version 4.2.4.

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.

New in version 4.2.5.

```
<?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>
```

2.4.6 Highlighting

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

Changed in version 4.2.4: 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>
   <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>
```

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.

Changed in version 4.2.4: 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.

Changed in version 3.2: 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>
```

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>
```

New in version 4.6: The `minCount` and `maxResults` parameters were added.

Facet exclusion

New in version 4.2.3.

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

New in version 4.1.

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

New in version 4.6.

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>
```

2.5 Caching

2.5.1 Search result caching

New in version 4.7.

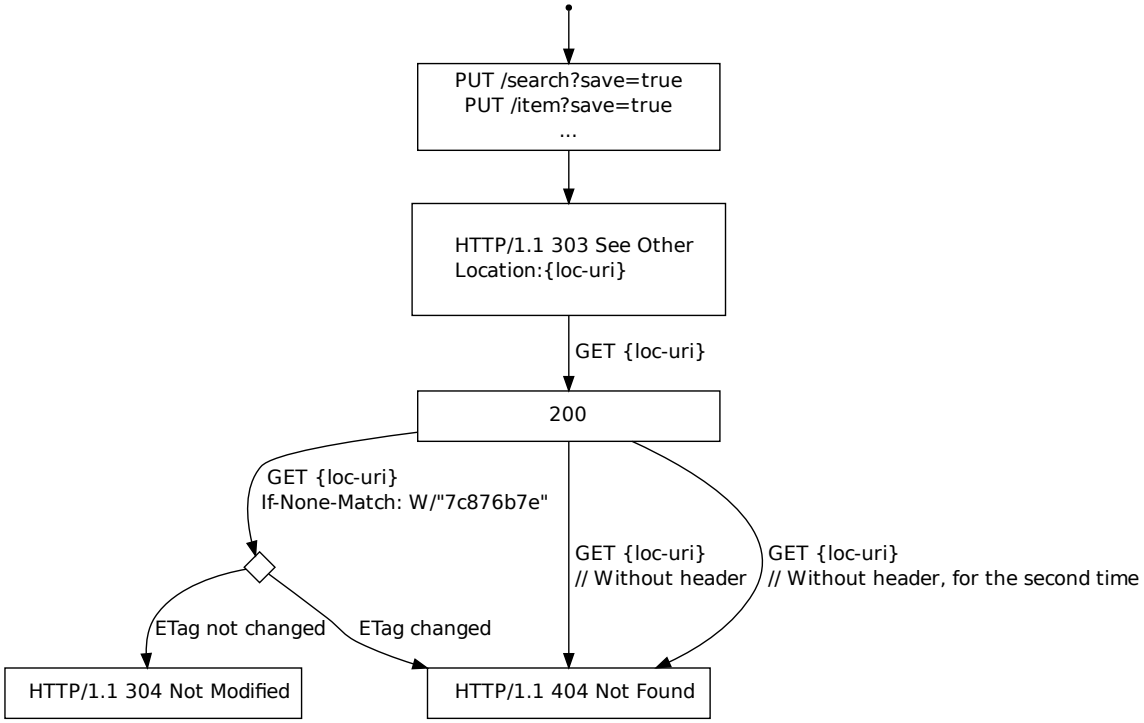
Saved searches

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

New in version 4.2.

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

New in version 4.2.

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>`.

Defining a rule

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. Vidispine is using Rhino (https://developer.mozilla.org/en-US/docs/Rhino) as the 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

The helper object contains some convenient functions for generating a new metadata object.

`helper.`**`createMetadata`**`()`
Returns a new *MetadataType*.

`helper.`**`createMetadataTimespan`**`(`*start*, *end*`)`
Returns a new *MetadataType*.Timespan.

Arguments

- **`start`** (*`string`*) – The start timecode.
- **`end`** (*`string`*) – The end timecode.

`helper.`**`createMetadataGroup`**`(`*name*`)`
Returns a new *MetadataGroupValueType*.

Arguments

- **`name`** (*`string`*) – The name of the group.

`helper.`**`generateMetadataField`**`(`*name*, *value*`)`
Returns a new *MetadataFieldValueType*.

Arguments

- **`name`** (*`string`*) – The name of the field.
- **`value`** (*`string`*) – The field value.

`helper.`**`metadataToStr`**`(`*metadata*`)`
Translate a metadata object to a string.

Arguments

- **`metadata`** – *MetadataType*

`helper.`**`log`**`(`*obj*`)`
Write the value of `obj.toString()` to the server log.

The following example script

```
var metadata = helper.createMetadata();
var timespan = helper.createMetadataTimespan("0", "100");
var group = helper.createMetadataGroup("mrk_marker");
var field1 = helper.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 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 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.

2.8.1 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. (New in 4.2.6.)

  0.0 left border

  1.0 right border
- `stl_yrelative`. Vertical position of base point relative to full video frame. (New in 4.2.6.)

  0.0 top border

  1.0 bottom border
- `stl_horizontalbase`. Horizontal position of base point relative to text bounding box. (New in 4.2.6.)

  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. (New in 4.2.6.)

  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`). (New in 4.4.1.)
- `stl_outline`. Type of outline. (New in 4.4.1.)

  (none) no outline

  `bar` rectangular outline

  `stroke` fat stroke around text
- `stl_outlinecolor`. Color of outline. (New in 4.4.1.)
- `stl_outlinesize`. Size (margin) of outline. (New in 4.4.1.)

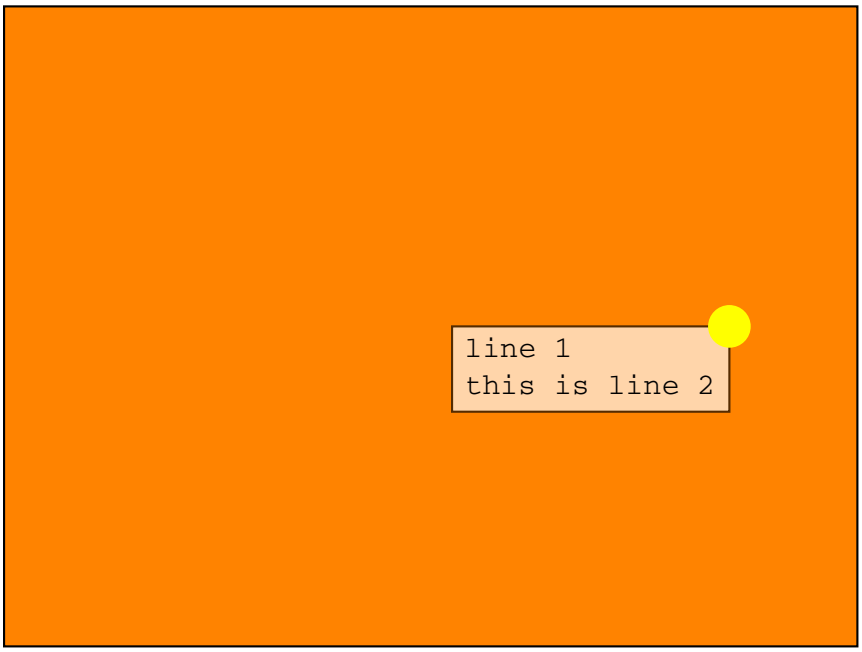
- `stl_font`. Font of subtitle. (New in 4.4.1.)

  `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>
```

2.8.2 Rendering subtitles in a sequence

New in version 4.0.

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.

2.8.3 SCC support

New in version 4.7.

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
```

2.8.4 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>
    <Change to 'iv' 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>
```

2.9 Examples

2.9.1 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>
```

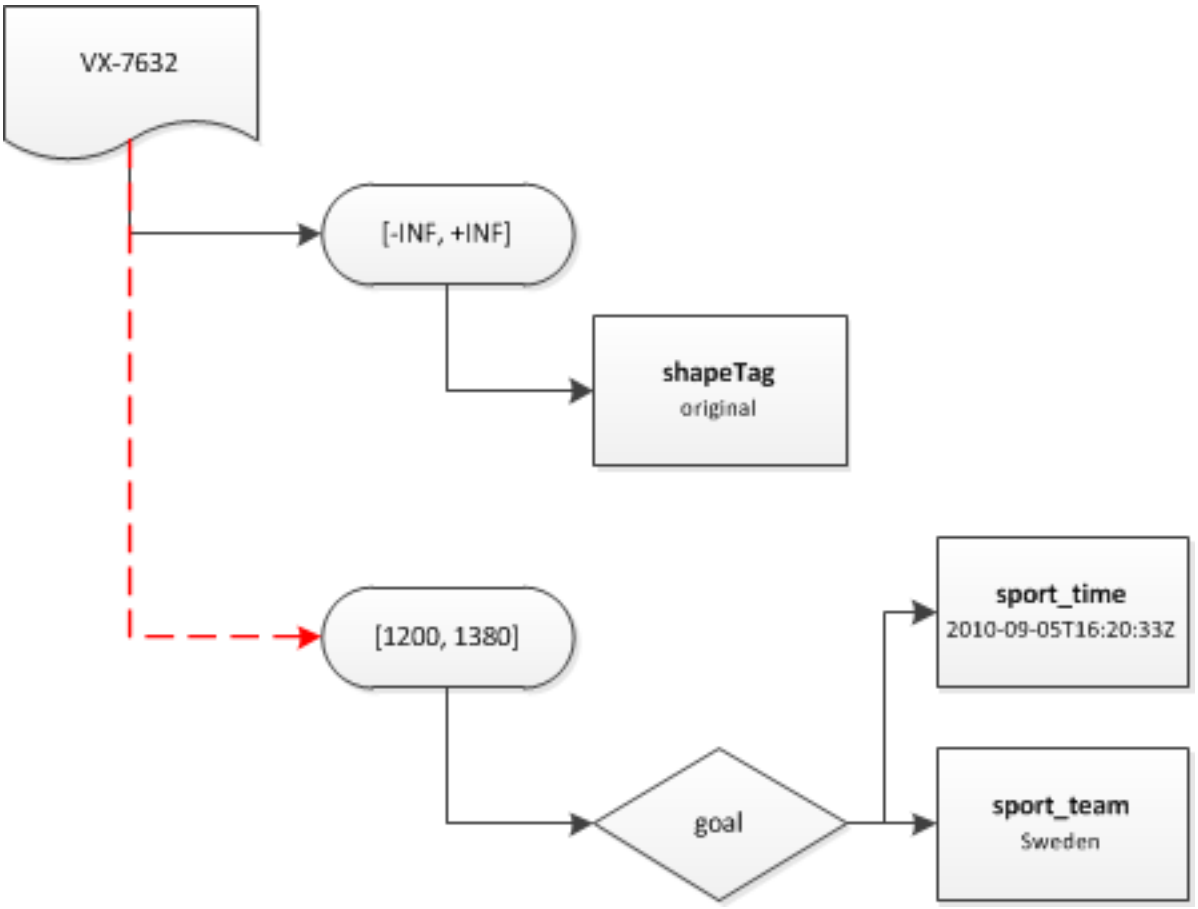
Modifying metadata

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:



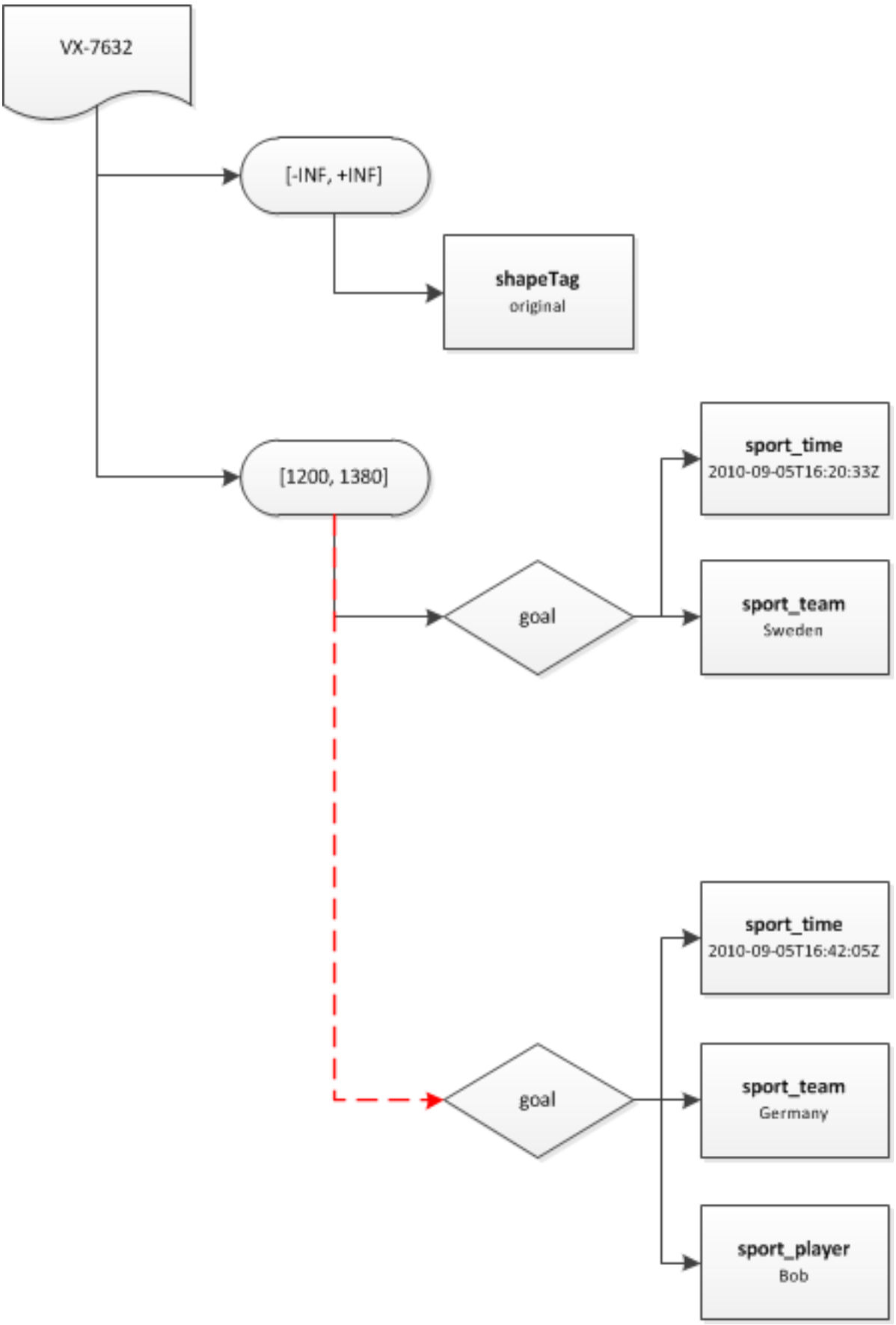

```
PUT /item/VX-7632/metadata
Content-Type: application/xml

<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

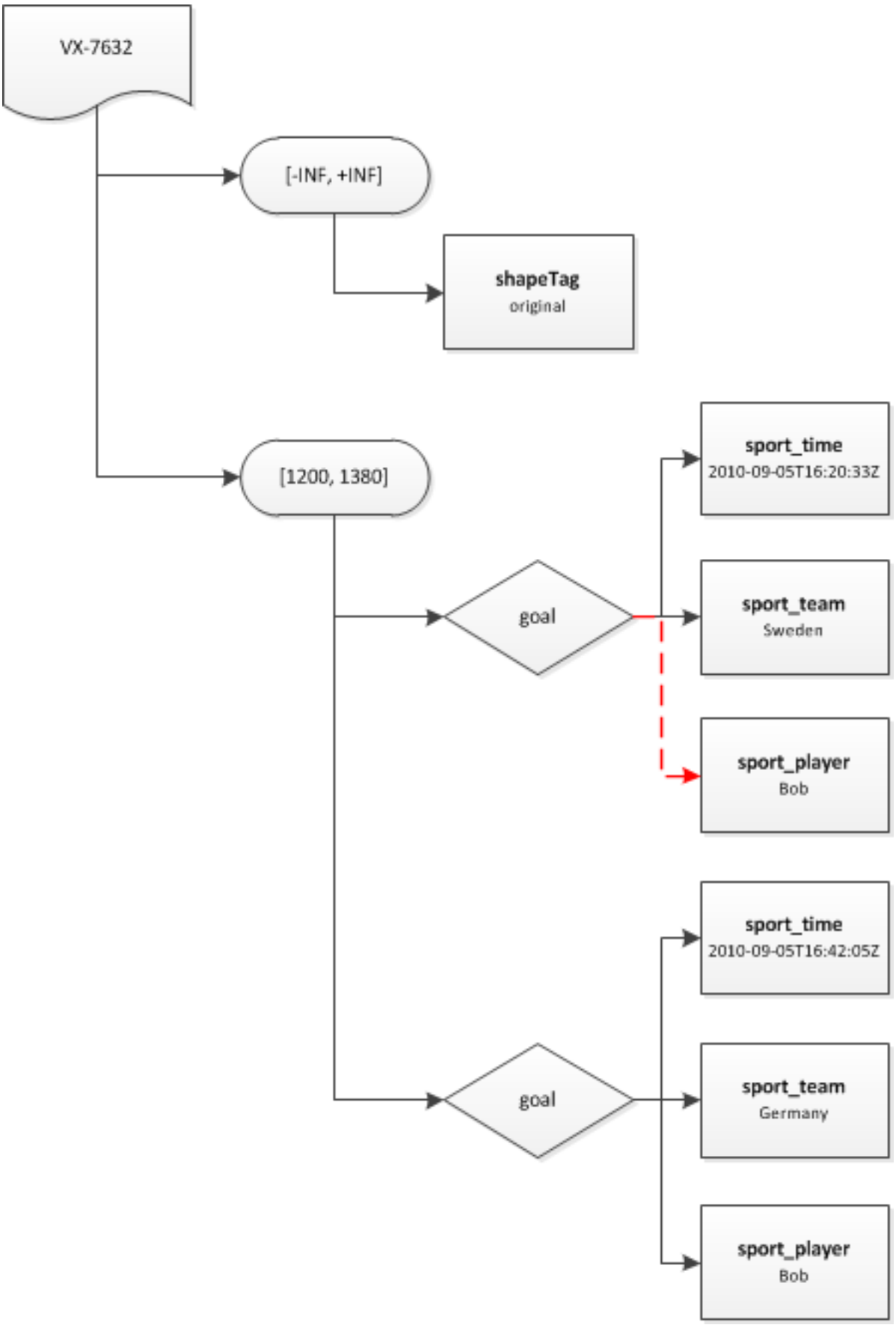
```
PUT /item/VX-7632/metadata
Content-Type: application/xml

<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]
sport_time
2010-09-05T16:20:33Z
goal
sport_team
Sweden
sport_player
Bob
sport_time
2010-09-05T16:42:05Z
goal
sport_team
Germany
sport_player
Bob

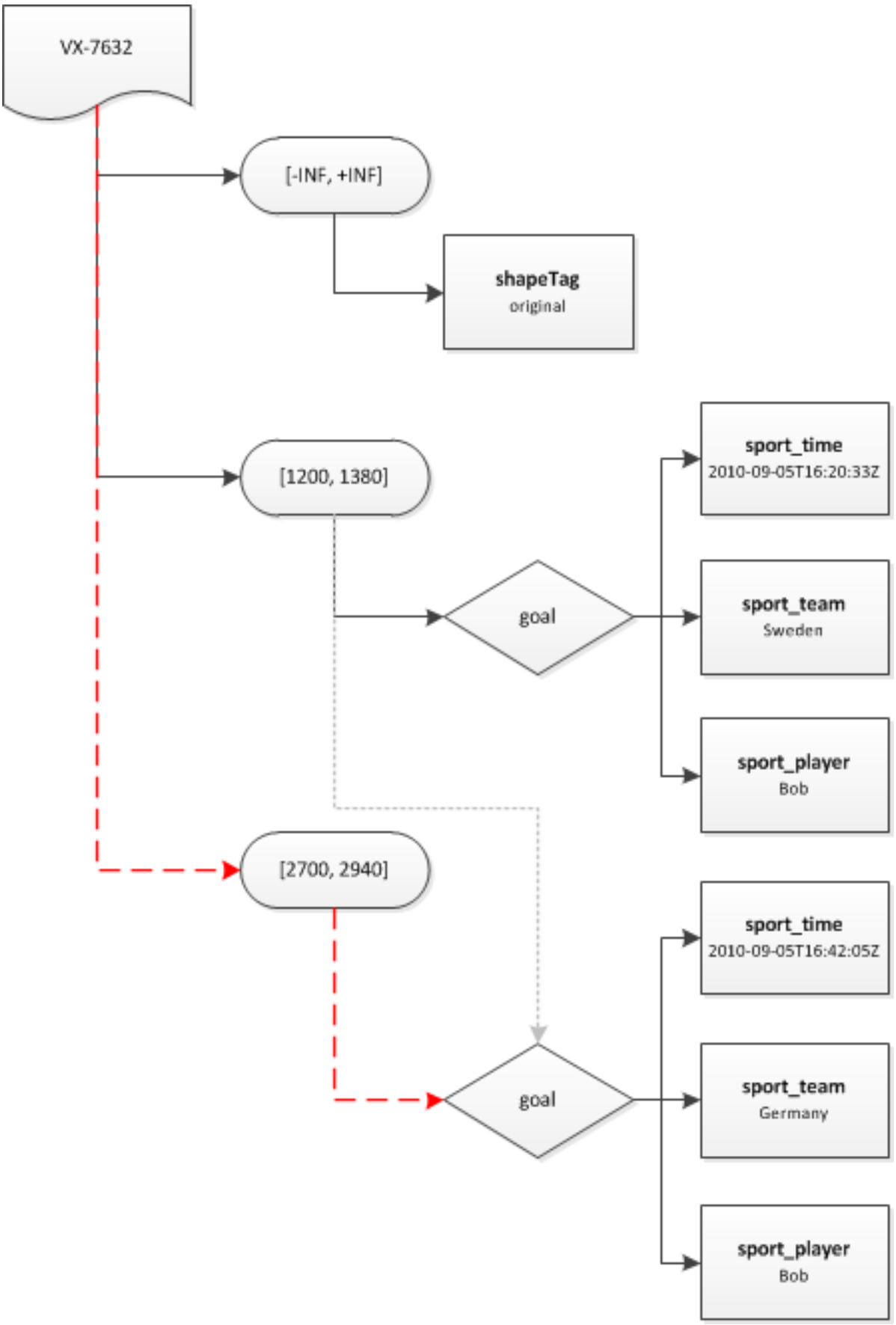
```
PUT /item/VX-7632/metadata
Content-Type: application/xml

<MetadataDocument xmlns="http://xml.vidispine.com/schema/vidispine">
   <timespan start="1200" end="1380">
      <group 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.

2.9.2 Defining a metadata schema

Based on the types in the *metadata example* we can specify a schema.

```
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>
```

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

New in version 4.2.4.

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

New in version 4.7.

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

New in version 4.7.

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
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 Metadata on collection to entity relations

New in version 4.7.

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>
```

3.1.7 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.

3.1.8 Representative thumbnails

New in version 4.0.

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>
   <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>
   <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
```

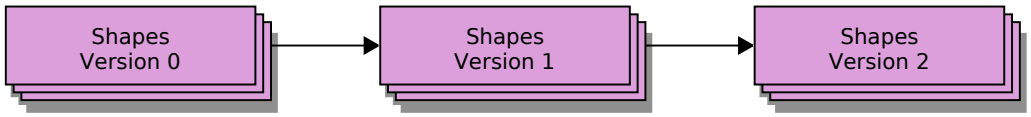
```
<JobDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <jobId>VX-169826</jobId>
  <user>admin</user>
```

```
  <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

New in version 4.0.3.

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

New in version 4.0.

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

New in version 4.0.

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

New in version 4.5.

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. Defaults to 0.5 (middle).
- `bitc_yRel`. Position relative to height of video. Defaults to 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. Defaults to `bar`.
- `bitc_outlinesize`. Size (margin) of outline.

Setting a maximum duration of a chunk in QuickTime files

New in version 4.1.

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

New in version 4.0.

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="0.5"/>
      <input channel="1" stream="1" gain="1.0"/>
      <input channel="3" stream="1" gain="1.0"/>
      <input channel="5" stream="1" gain="1.0"/>
    </mix>
    <mix>
      <input channel="1" stream="1" gain="1.5"/>
      <input channel="2" stream="1" gain="1.0"/>
      <input channel="4" stream="1" gain="1.0"/>
      <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

New in version 4.1.

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

New in version 4.1.1.

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>
```

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. (New in 4.4.)

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

New in version 4.7.

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>
```

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.

Common settings for video and image

thumbnailformat

By default, video thumbnails are JPEG and image thumbnails and PNG.

New in version 4.2.12.

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`.

New in version 4.5.

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`).

New in version 4.5.

The transcoder now comes preinstalled with the most common profiles.

New in version 4.5.

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>
```

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

New in version 4.4.3.

If `true`, sharpens the image. May produce better results after scaling.

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 H264

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>
```

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

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>
```

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`.

4.3.8 RED

New in version 4.1.

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

New in version 4.2.

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>
  <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>
```

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 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.

New in version 4.1: 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)).

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 New in version 4.2.9.

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.

New in version 4.2.5: 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.

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` storage metadata. 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

New in version 4.4.

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 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

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

New in version 4.0.

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

New in version 4.2.3.

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.
- With username and password credentials, the file should be a directory, containing two files, `username` and `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
```

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.

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

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/`

New in version 4.1.2: 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`.

sftp

Syntax `sftp://{user}:{password}@{host}/{path}`

Example `sftp://johndoe:secr3t@example.com/mystorage/`

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).

Changed in version 4.6: The default locations specified by the AWS SDK are now also searched.

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.

New in version 4.4.

`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.

New in version 4.4.

`signer` The algorithm to use to signing requests. Valid values include `S3SignerType` for AWS signature v2, and `AWSS3V4SignerType` for AWS signature v4.

New in version 4.5.3.

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`

New in version 4.0.3.

Changed in version 4.5: Support for infrequent access was added.

`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`

New in version 4.4.1.

`ds3`

Syntax `ds3://{accessKey}:{secretKey}@{bucket}/{path}`

Example `ds3://KDASODSALSDI8U:RxZYlu23NDSIN2Nyq@bucketname/?endpoint=http://blackpearl-e`

Note Spectra BlackPearl Deep Storage Gateway.

New in version 4.5.

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/`

New in version 4.0.1.

gs

Google Cloud Storage (https://cloud.google.com/storage/).

New in version 4.7.

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/`

See also:

See `here` for some notes on how to write URIs.

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 Setting a user for jobs started as a result of an auto import rule

New in version 4.6.

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>` tag in the rule XML:

```
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>
```

5.2.2 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.

5.2.3 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.

5.2.4 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`.

5.2.5 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*.

New in version 4.1.1: 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.

New in version 4.2.2: Storage rules on collections can now be inherited to items in sub-collections. See *Inherited rule example*.

New in version 4.6: Support for load balancing files across multiple storages was added.

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

New in version 4.6.

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

New in version 4.2.2.

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.getChannel()`
: Most of the time this will return null, except when you want to *split audio channels to separate files*.

  New in version 4.1.

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

New in version 4.6.

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:

- `How to write VS URI's`
- The URLEncode and URLDecode Page (http://www.albionresearch.com/misc/urlencode.php)

CHAPTER

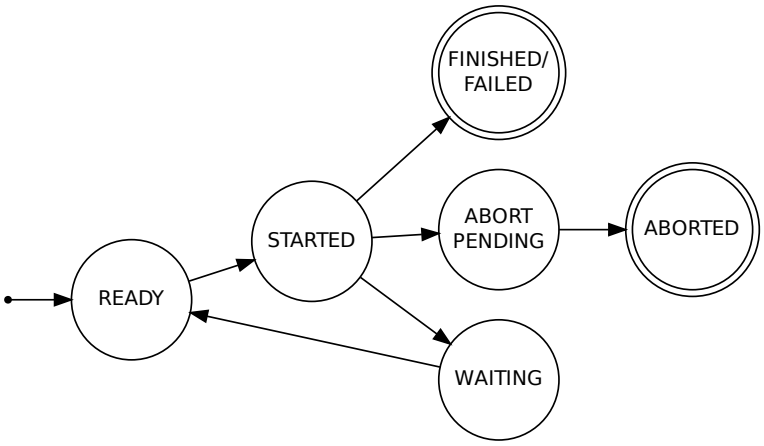
SIX

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

New in version 4.2.2.

Using job pools it is 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 running. Job pools are configured using the *job pool configuration resource*.

```
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>
```

Note that the max concurrent job setting will only have an effect if it is lower then the size of all pools combined.

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>
```

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 an EJB 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

New in version 4.6.

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 Rhino (https://developer.mozilla.org/en-US/docs/Rhino). 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.

  New in version 4.2.2.

`job.`**`log`**`(`*description*`)`
: Logs a message related to the current job step.

  New in version 4.2.2.

`job.`**`getData`**`(`*key*`)`
: Gets the data for the given key.

`job.`**`setData`**`(`*key*, *value*`)`
: Sets the data for the given key.

`job.`**`fail`**`(`*errorMessage*`)`
: Fails the current step, but the step will be retried (up to five times).

`job.`**`fatalFail`**`(`*errorMessage*`)`
: Fails the current step and job.

6.2.2 Pausing job execution

New in version 4.0.

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*. 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.

Note: (New in 4.1.5.). Before version 4.1.5, *job.fail* should be used to signal that the job should keep waiting. Since 4.1.5, both *job.fail* and *job.wait* can be used.

`job.`**`wait`**`(`*reason*`)`
Sets the job in WAITING state.

Arguments

- **reason** (*string*) – An explanation of what the job is waiting for.

Example

```
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
  ....
}
```

6.2.3 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;
```

6.2.4 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

New in version 4.4.

Task groups can be used to control the transcoders that a specific job should use. 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 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.

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 an EJB method.
- Send a JMS message.

- Execute a JavaScript.
- Send a message to Amazon SQS (New in 4.7.)

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

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.

7.5 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.

Changed in version 4.4: 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.

Warning: Do not connect multiple Vidispine installations to the same set of transcoders unless each installation has a distinct site name. If installations have the same site name then jobs from one installation may conflict with jobs from another.

Changed in version 4.4: This no longer applies; see above notice.

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 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

New in version 4.2.3.

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

New in version 4.2.3.

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

Since 4.2.5

Controls where temporary files are stored. Defaults to `/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. (New in 4.2.3.) Since version 4.2.3, this setting is used with more formats then before.

```
<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

New in version 4.5.

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>
```

(New in 4.2.5.) 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.

8.1.5 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>
```

For GlassFish, a new HTTP listener (`noauth-listener`, default port is the standard API port + 9) is installed on GlassFish for communication from the transcoder to GlassFish. The rules for how the address to GlassFish 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.
3. If the port for `noauth-listener` can be determined (GlassFish only), it is used with the reverse address.
4. If the port for `http-listener-1` can be determined (GlassFish only), it is used with the reverse address.

For JBoss, it is currently required that at least `apiNoauthPort` is set.

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

New in version 4.4.

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.

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.

Changed in version 4.5: Any method supported by Vidispine can now be specified as the source or destination, meaning that the watch folders no longer 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 remaing unchanged for 3 checks.

Changed in version 4.4.2: The interval and checks settings were added. Files may now also continue growing past the timeout.

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 directory on the file system where the thumbnail database files will be stored.

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

New in version 4.6.

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 (New in 4.2.2.).

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

New in version 4.2.2.

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:

```
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 start vidispine-agent
$ 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

New in version 4.6.

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`: Vidipsine 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 Vidsipine server to connect to VSA

2. Restart VSA:

```
$ sudo 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 (*Retrieve list of 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.

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. `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, each with a number of segments. 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>
```

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	-Inf–Inf	Vertical offset in output in percentage.
	horiz	-Inf–Inf	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>
```

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>
```

CHAPTER

TEN

USERS, GROUPS, AND ACCESS CONTROL

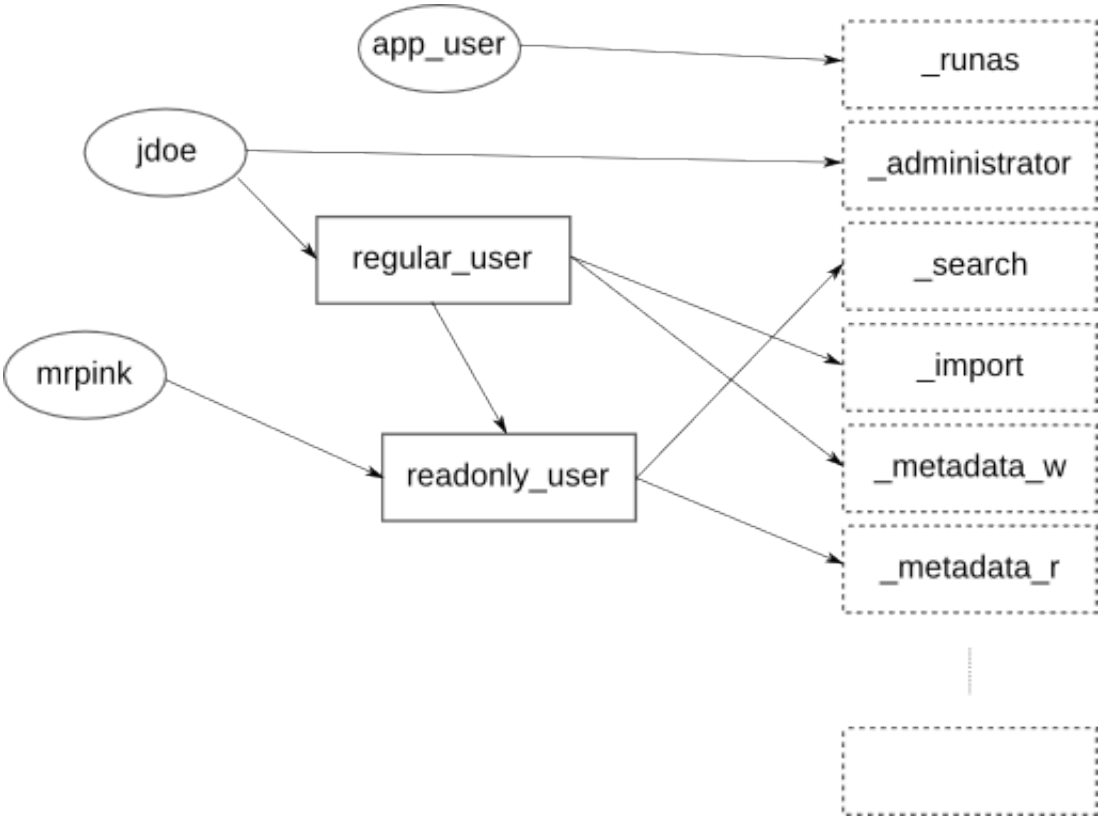
The user management system in Vidispine consists of users, groups, and roles.

- 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 will use 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.

```
<AccessControlListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <access id="VX-21">
    <loc>http://vs.example.com:8080/API/collection/VX-16/access/VX-21</loc>
    <recursive>true</recursive>
    <permission>OWNER</permission>
    <user>admin</user>
  </access>
</AccessControlListDocument>
```

Here the admin user has created a collection, and is thus the owner with full access. Access controls from collections, and libraries, are inherited to the entities in them. The `recursive` element can be used to control if the access control should apply to the child items or not.

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. The rule of thumb here is that if there's a matching access control entry set on the item then that applies otherwise the item's ancestor collections and libraries are traversed. Then the collection or library that grants the **lowest** access will apply. If no matching access control entry can be found, access will be denied.

Furthermore an access control entry that is more specific take precedence over an entry that is less specific. If two entries are determined equally specific then the entry that grants the highest access applies. An example of this is that an entry that restricts access for an items entire metadata is considered less specific than one that only restricts access for a certain field in the metadata.

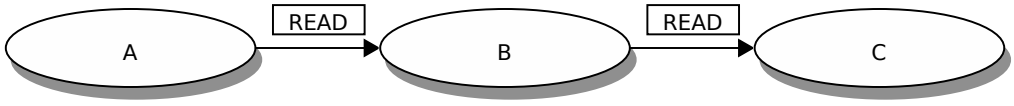
The above can be illustrated by the priority list below:

1. Controls with a high explicit priority take precedence over controls with lower explicit priority.
2. Controls directly on the item take precedence over controls on ancestor collections and libraries.
3. Controls that describe specific users take precedence over controls that describe groups.
4. Controls that are more specific take precedence over controls that are less specific.
5. If directly applied to an item:
    - Controls that grant more access take precedence over controls that give less access.
6. If applied to an ancestor collection or library:
    - Controls that grant less access take precedence over controls that give more access.

An explicit priority can be assigned by setting the `priority` element in the AccessControlDocument to the desired level. The default is 0. 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.

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.



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.

New in version 4.1.

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.

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 *Get an authentication token for a user*. This token can then be used in subsequent calls as credential by specifying the http header:

```
Authorization: token {token}
```

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 `/user/{user-name}/validate`.
3. Store the user name with the session.
4. Use the `RunAs` header with all communication to the Vidispine API.

10.4.2 Apache Shiro Integration

New in version 4.1.

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
4.1	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`. On GlassFish installations the instance root folder is typically `glassfish/domains/domain1/`.

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. On GlassFish the JAR file should be copied to `glassfish/lib/`.
2. Create a `shiro.ini` file based on the above template and modify it to your needs.
3. Start/Restart the application server.

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.

OAuth 2.0

New in version 4.4.

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 static 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

To set a static public key, add the following to shiro.ini:

```
[main]
...
oauth2Auth = com.vidispine.security.auth.BearerAuthenticationFilter
oauth2Auth.x509Certificate = {x509-certificate}
oauth2Auth.tokenUser = email  # example
...

[urls]
/** = noSessionCreation, tokenAuth[permissive], oauth2Auth[permissive], authcBasic
```

Where `{x509-certificate}` is an X.509 certificate encoded with Base64, e.g. `MII...==`. Vidispine will use this certificate and verify that the token's signature matches the certificate. Then, the tokenUser property is used to describe which key in the token's JSON object that should be used as the 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 the example above with Google, the https://www.googleapis.com/auth/userinfo.email scope is used.

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).

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.

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 /self-test/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

MONITORING

To get better insight into the operations of jobs and services you can collect metrics into 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.

New in version 4.2.3.

12.1 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.

12.1.1 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>
```

12.1.2 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>
```

12.2 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/).

12.3 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.

12.3.1 Indexing

- Meters:
  - `reindex.{index}.started`

  - `reindex.{index}.finished`
  - `indexer.solr.request.failed`
- Timers:
  - `indexer.solr.update.time`
  - `indexer.solr.delete.time`
  - `indexer.solr.commit.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.

    New in version 4.7.

12.3.2 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`.

12.3.3 Solr

- Meters:
  - `solr.request.failed`
- Timers:
  - `solr.query.time`
  - `solr.update.time`
  - `solr.commit.soft.time`
  - `solr.commit.hard.time`

  - `solr.optimize.time`

12.3.4 Storage

- Meters:
  - `storage.file.found`
  - `storage.file.changed`
  - `storage.file.deleted`
  - `storage.file.hashed`
  - `storage.file.checksum.bytes.read`
  - `storage.fs.stat`
    * The number of `stat` call made.

12.3.5 Transfer

- Meters:
  - `transfer.bytes.transferred`
  - `transfer.started`
  - `transfer.finished`
  - `transfer.finished-part`
  - `transfer.failed`
  - `transfer.blocked`

12.3.6 Service

- Meters:
  - `service.exception`
- Gauges:
  - `service.load.5`
    * The 5 minute load.
  - `service.load.60`
    * The 60 minute load.

12.3.7 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

12.3.8 Broker

- Gauges
  - broker.queue.{queue}.size

    The size of a specific queue. Note that this metric is only present when using the embedded broker.

    New in version 4.7.

CHAPTER

THIRTEEN

CONFIGURATION AND INTEGRATION

13.1 System configuration

13.1.1 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>
```

13.1.2 Metrics configuration

See *StatsD* on how to configure how metrics are sent to StatsD. The configuration resource is described at *Metrics settings*.

13.1.3 FTP pool configuration

New in version 4.2.4.

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*.

13.1.4 Database purging

Vidispine supports mechanisms for purging old information in database tables. Especially three tables can grow quite large without purging enabled.

Change-log table

New in version 4.2.4.

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.

Audit trail table

New in version 4.2.4.

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` will be removed and put in a file inside the `auditTrailPurgingDirectory` folder.

A sensible value is 43200 or higher, corresponding to one month.

Job table

New in version 4.3.

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` will be removed and put in a file inside the `jobPurgingDirectory` folder.

13.1.5 Configuration properties

Configuration properties are used in Vidispine to control system-wide parameters.

General

`apiUri`
URI to Application Server. Used by transcoder(s), so need to be 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 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`

Since 4.4

`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`

Since 4.0.4

`validatexml`
Enable schema validation of the incoming and outgoing xml document.

Default `false`

Since 4.2.1

`slaveLicenseProxy`
Use a proxy for *Connection to Vidispine master license service*. Format is

- `http://IP:port/` or
- `socks://IP:port/`

Proxy authentication is not supported.

Default none

Since 4.2.5

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/`

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`

Since 4.1

solrCollection
For SolrCloud: The collection in Solr to be used by Vidispine.

Mandatory No (Yes for SolrCloud)

Example `collection1`

Since 4.1

solrQueryTimeout
The request timeout in milliseconds to use when querying Solr.

Default `60000`

Since 4.1.1

solrPingAttempts
The number of times to ping a Solr node before aborting an active request.

Default `5`

Since 4.1.1

solrPingTimeout
The request timeout in milliseconds to use when checking if a Solr node. is alive

Default `5000`

Since 4.1.1

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 sending soft commit to Solr automatically.

Default `true`

`solrUpdateQueueSize`
Number of documents Vidispine will send in batch to Solr.

Default `100`

`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 *Ordering collections*.

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.

Changed in version 4.4: This is now disabled by default.

Default `false`

Since 4.2.8

`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`

Since 4.2.13

`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`

Metadata

`disableMetadataSchema`
If a metadata schema has been defined (see *Metadata schema*), allows metadata that does not comply to the schema.

Default `false`

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`

Since 4.2.2

userTokenDefaultInterval
Default token expiration time, in seconds.

Default `60`

Since 4.2.2

userTokenRefreshInterval
Minimum time between token refreshments, in seconds.

Default `10`

Since 4.2.2

Jobs and imports

concurrentJobs
Number of *jobs* that are allowed to be started.

Default `3`

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`

Since 4.2.11

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`

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`

Since 4.0.10

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`

Since 4.0.3

alwaysGenerateThumbnails
When `true`, thumbnails will be generated on import even if no transcoding takes place.

Default `false`

Since 4.0.3

mediaCheckInterval
The retry interval of media check (seconds).

Default `3`

Since 4.1

Storage and file

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`

Since 4.1

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`

Since 4.2.5

fileHashAlgorithm
Hashing algorithm used. If changed, the `c_hash` column of the `t_file` table should probably be set to `NULL`.

Example `SHA-1`

enableTranscoderHashing
Off-load file hash calculation available transcoder.

Default `false`

Since 4.2.4

fileTempKeyDuration
Number of minutes a no-auth URI is valid (*Auto method types*).

Example `10`

useS3Proxy
When `true`, Vidispine will create S3 pre-signed URLs for reading during job.

Example `false`

Since 4.1

s3ProxyValidTime
The validate time (in minutes) of S3 pre-signed URL.

Example `60`

Since 4.1

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.

Default `5242880`

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`

Since 4.1

azureSasValidTime
Specifies for how many minutes an AZURE-SAS URI will be valid. See *Get status of file in storage*.

Example `60`

Since 4.0.1

`stornextFileMetadata`
Specifies which fields that should be stored on the Vidispine file entity from StorNext metadata. See *StorNext Metadata*.

Default `location,class,existingCopies,targetCopies`

Since 4.2.3

`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`

Since 4.2.3

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`

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`

`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*).

Since 4.1.1

`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`

Since 4.2.5

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.

Default `false`

Since 4.2.4

Library

libraryUpdateInterval
Default library update interval in the system (seconds).

Default `60`

Since 4.1

libraryExpireTime
Default library expire time in the system (seconds).

Default `86400`

Since 4.1

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`

Since 4.0

Growing files

fileGrowingTimeout
The max time a file can keep growing (seconds).

Default `36000`

Since 4.1

`fileNotGrowingTimeout`
A file is considered as not growing if it has not been changed during this period (seconds).

Default `600`

Since 4.1

Services

`itemDeleteInterval`
The running interval (seconds) of `ItemDeleteCruncher` during the "idle" period (no item to delete).

Default `60`

Since 4.1

`itemDeleteIntervalShort`
The running interval (seconds) of `ItemDeleteCruncher` during the "busy" period (there are items to be deleted).

Default `5`

Since 4.1

`itemDeleteExecutionTime`
Max running time (seconds) of `ItemDeleteCruncher` thread, after that it goes to sleep.

Default `5`

Since 4.1

`fileHashExecutionTime`
Max running time (seconds) of a file hashing thread, after that it goes to sleep.

Default `10`

Since 4.1

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.

Since 4.2.2

Default `true`

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*.

Since 4.2.4

Default not set

auditTrailPurgingDirectory

Since 4.2.4

Default not set

changeLogPurgingTime
Remove all processed change-log entries older than the specified time (in minutes). See *Change-log table*.

Since 4.2.4

Default not set

changeLogForcePurgingTime
Remove all change-log entries (processed or unprocessed) older than the specified time (in minutes).

Since 4.2.4

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*.

Since 4.3

Default not set

jobPurgingDirectory

Since 4.3

Default not set

disableSequenceChecker

Since 4.6.1

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.

Since 4.4.1

Default 60 seconds

bulkyMetadataKeysToIgnore
Comma-separated list of bulky metadata keys to ignore from analysis results, e.g. `crop`,

Since 4.4.1

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.

Since 4.4.1

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.

Since 4.6

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.

Since 4.6

Default true

13.1.6 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}`

Since 4.3

com.vidispine.license.tmpdir
The directory where temporary license files may be stored.

Default `${com.sun.aas.instanceRoot}`

Since 4.3

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}`

Since 4.3

com.vidispine.log.dir
The directory containing the server log files.

Default `${com.sun.aas.instanceRoot}/logs`

Since 4.3

vidispine.identifier.format
If `full`, output *Long identifiers*.

Default Normal, short identifiers.

13.2 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. 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*.

13.2.1 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.

13.2.2 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>
```

13.3 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 can be obtained here: http://www.vidispine.com/license

13.3.1 How it works

The license is a physical file which must reside in the Java application server domain folder. The file is either called `License.lic` or `slaveAuth.lic` depending on if the licensing is standalone/master or a slave setup.

The Vidispine license is tied to your systems MAC address(es) (http://en.wikipedia.org/wiki/Mac_address). For Cloud-based systems with non-persistent MAC-address allocation (such as Microsoft Azure), a master-slave license setup can be used to ensure proper licensing of a Cloud based Vidispine instance.

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.

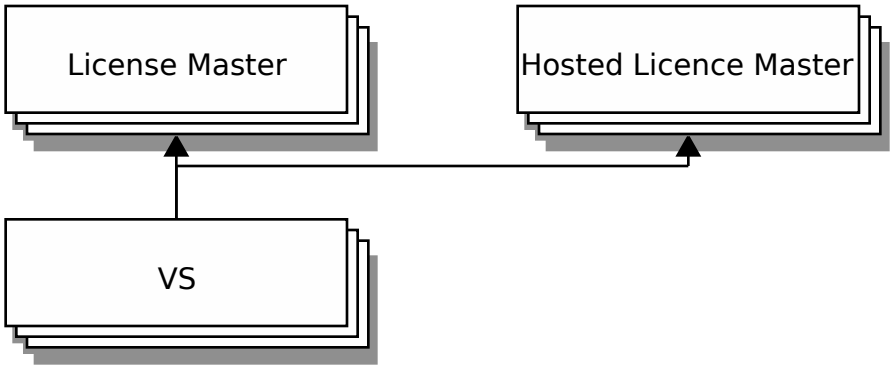
Deployment license When installing Vidispine using the installer, a non-MAC bound deployment license will be installed. This license will allow you to verify that your system was properly installed. The license will allow for 2 users, 100 assets, 1 transcoder, 1 storage area, and encoding/decoding of 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.

13.3.2 Master-slave licensing



(Also called Cloud licensing)

This license model was introduced in order to license Vidispine instances running in environments with non-persistent MAC-address allocation, such as on virtual hosts, Microsoft Azure etc. The model allows you to set up your own license master to sublicense your Vidispine instances, i.e. slaves. Vidispine also offers a hosted master license service for such environments.

Note: The terms *master* and *slave* only refer to how the licensing is policed and has no relation to the actual role of your Vidispine instance.

Vidispine hosted license service

Vidispine offers a redundant online master license service for licensing of Vidispine systems in environments with non-persistent MAC-addresses. All you need is a `slaveAuth.lic` file for your server - contact us at license@vidispine.com for further information.

Connection to Vidispine master license service

The connection to the Vidispine master license service 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*. (New in 4.2.5.)

Configure your own master-slave setup

In order to set up your own license master, the master node(s) must reside on a server with a persistent MAC-address.

You need the following files:

1. A master license key (`License.lic`) to license your master node.
2. A slave license key (`Slave.lic`) to be installed on your master node (via the API) in order to allow slaves to connect to the master.
3. A slave connection string file (`slaveAuth.lic`) on the slave node(s) in order for the slave(s) to find the master.

The master and slave licenses are paired together by the property `MasterIdentifier`, which needs to be equal in both files. The slave license and the slave connection string files are paired together by the property `SlaveIdentifier`, which needs to be identical in both files.

Your master node needs to run the full Vidispine middleware (API) to enable the license master service.

Contact us at license@vidispine.com to obtain a master & slave license key pair.

Install master license on master node

Copy the master license key `License.lic` to your Java application server domain root folder on the master node.

License validity and status can be seen in the response from *GET /version*.

Install slave license on master node

Copy the slave license key `Slave.lic` (corresponding to the master key) to a convenient location on the master node, such as the Java application server domain root folder (do not rename the slave license file to `License.lic`, as this will overwrite the master node license file).

Make a *PUT /license/slave* request on the master node to install the slave license:.

Example

Using GlassFish, if the slave license is called `Slave.lic` and is located in the domain root, the request should be:

```
$ curl -X PUT -uadmin:admin http://localhost:8080/API/license/slave?path=Slave.lic
```

Install slave connection string file on slave (manual)

Create a file named `slaveAuth.lic` containing the `SlaveIdentifier` and IP-numbers of your master node(s). Copy the file to your Java application server domain root folder on the slave node.

The `SlaveIdentifier` property value must be equal to the `SlaveIdentifier` value in your slave license key installed on the master. The `MasterIP` property value must equal the IP numbers or DNS addresses of your master node(s).

Example

```
SlaveIdentifier=your-slave-id
MasterIP=http://192.168.0.1:8080/, http://my.other.server:8080/
```

Multiple IPs (comma separated) are supported for redundancy. A slave will automatically fail over to the secondary master if the primary master goes down, and vice versa.

License validity and status can be seen from *GET /version*.

Install or update slave connection string (via the API)

The slave connection string can also be installed using *PUT /APInoauth/license/auth-info*. For 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*.

13.3.3 Slave management and monitoring

See *Slave management and monitoring* on how to manage and monitor the connected slaves.

13.3.4 Redundancy and timeouts

A heartbeat request is performed from the slaves towards the master every 60 seconds. A slave will automatically fail over to the secondary master if the primary master goes down, and vice versa. If the slave does not receive a valid license from any master within 180 minutes, the slave goes into read-only-mode and it's entry will be deleted from the list of slaves on the masters. A request to *GET /version* on the slave will report "License status invalid". The slave will go back to normal operation as soon as any of the masters become available again.

13.4 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.

13.4.1 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.

`api.`**`queryParam`**(*key*, *value*)
Adds a query parameter to the API URI.

`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

- **`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.

`api.`**`timeout`**(*timeout*)
Sets the timeout of the request.

Arguments

- **`timeout`** (*`int`*) – The timeout in milliseconds.

New in version 4.0.3.

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`.

New in version 4.0.3.

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`**`()`

New in version 4.0.3.

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.
- `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

New in version 4.0.3.

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.

Example:

```
var uri = api.path('version').getInfo().uri;
http.uri(uri).user('admin','admin').dataType('JSON').get().licenseInfo.licenceType
```

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` - An optional timeout in milliseconds.
  - `input` - Optional input to send to standard input.
  - `output` - Optional `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` - Optional `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

New in version 4.0.3.

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** – 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.

13.4.2 Debugging JavaScript

New in version 4.0.3.

The JavaScript code can be debugged using Eclipse. 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.

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. To test, POST (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.5) some JavaScript code to `API/javascript/test`, e.g. using curl:

   ```
   $ curl -uadmin:admin -Hcontent-type:application/javascript \
   localhost:8080/API/javascript/test -X POST --data-binary \
   '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.

13.4.3 Interfacing with the JavaScript engine manually

New in version 4.0.3.

In order to test functionality, the JavaScript engine can be called manually. For more information, see *JavaScript*.

13.4.4 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);'
```

13.5 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.

13.5.1 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;
    }
  }
};
```

13.5.2 Amazon Glacier

New in version 4.1.1.

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>glacierVaultName</key>
      <value>{vault name}</value>
    </field>
    <field>
      <key>glacierEndpoint</key>
      <value>https://glacier.us-east-1.amazonaws.com/</value>
    </field>
  </metadata>
</StorageDocument>
```

Files can then be moved to this storage either using storage rules or by initiating a copy job (*Create a file move/copy job*). 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.

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

Query Parameters

- **extraData** – `expirationInDays={number-of-days}`

Produces

- **text/plain** – Informational text.

This will cause Vidispine to ask Glacier to initiate a restore. Once the restore is complete, the file will be put in the CLOSED state, and it is available for direct access.

The `extraData` parameter can be used to specify that the restored files should be available for a limited time, in this example for five days. Once it has expired, it will be removed from direct access and once again end up in the ARCHIVED state. If no value is provided, a default of -1 will be used.

13.5.3 Front Porch Diva Integration

New in version 4.1.

Requirements

- StoragePlugin.jar

Installation

Deploy StoragePlugin.jar in GlassFish:

- Open the web administration console.
- Navigate to *Applications*.
- Click *Deploy...* and select the StoragePlugin.jar file.
- Make sure that *Type* is set to *EJB Jar* and that *Compatibility* is selected.
- Click *OK*.

Configuration

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>
```

Usage

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.

13.6 S3 Event Notifications

New in version 4.4.

Vidispine supports polling Amazon SQS for S3 events, to detected changes on the Amazon bucket. This makes it less expensive to detect file changes on S3 buckets.

13.6.1 Configure S3 notifications

1. Create an Amazon SQS queue. You could create the queue from either the AWS Management Console (http://console.aws.amazon.com/) or using the SQS SDK. Make sure that the S3 bucket is allowed to send messages to the queue, for example, using a policy like:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
```

```
            "Action": [
                "sqs:*"
            ],
            "Resource": [
                "arn:aws:sqs:eu-west-1:12121221212:exampleQueue"
            ],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:s3:*:*:bucketname"
                    ]
                }
            }
        }
    ]
}
```

Note:

- Use one queue per bucket. Don't send events from multiple buckets to the same queue, as this is not supported by Vidispine.
- Vidispine will connect to the queue using the credentials from the S3 method URI, so that user must have access to both the bucket and SQS queue.

2. Configure the S3 bucket to send *ObjectCreated:** and *ObjectRemoved:** events to SQS. The notifications can be set up from the S3 bucket properties in the AWS Management Console, or using the S3 SDK.

13.6.2 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
```

```
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>
    </metadata>
  </method>
  ...
</StorageDocument>
```

2. Then make sure that the storage metadata *refreshOnStart* 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. You can then set the storage to be scanned less often by setting *refreshInterval* to a larger number than default, for example 1 hour.

   Every *refreshInterval*, Vidispine will perform a full scan, as a precaution of any possible inconsistency.

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>
```

13.7 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.

13.7.1 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.

13.7.2 Storage configuration

The following metadata field must be set on the source and destination storage (*Key-value metadata*).

`signiantAgent`

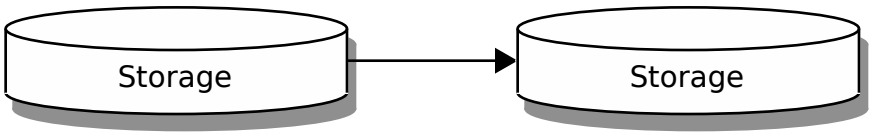
The name of the agent connected to this storage. This can also be set to an agent group using the format `<group_name>!<organization_number>`.

Storage methods

Each Signiant enabled storage needs a `file:/` storage method. This will be used to determine the source and destination paths for transfers.

13.8 Aspera Integration

Vidispine can initiate transfers between storages using Aspera. For this to work, the source and destination storages must first be configured properly.



13.8.1 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.

13.8.2 Destination storage configuration

`asperaDestinationUri` The Aspera URI corresponding to the root folder of this storage

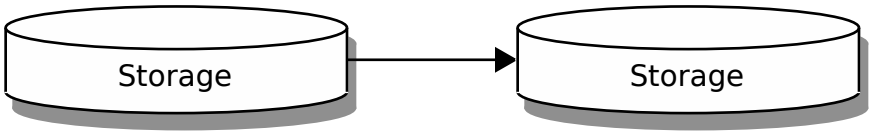
Example `fasp://10.18.12.11:22/usr/local/aspera-destination?user=username&passwo`

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.

13.9 FileCatalyst Integration

New in version 4.2.3.

Vidispine can initiate transfers between storages using FileCatalyst. For this to work, the source and/or destination storages must first be configured properly.



13.9.1 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.

13.9.2 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>
```

13.10 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.

13.10.1 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
```

13.10.2 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.)

13.11 EVS IP Director Integration

New in version 4.1.

It is possible to map data inside "log info" (<Log>) in a EVS metadata file to Vidispine metadata.

13.11.1 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.

13.12 StorNext Integration

New in version 4.2.3.

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.

13.12.1 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>
```

13.12.2 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.

13.13 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 (New in 4.0.), PLACEHOLDER_IMPORT (New in 4.0.3.), ESSENCE_VERSION (New in 4.0.3.) and AUTO_IMPORT (New in 4.2.8.) are supported.

13.13.1 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.

13.13.2 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'
```

(New in 4.2.8.) 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.

New in version 4.2.8.

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'
```

13.13.3 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?&jobmediafile=115&
↪frame=467&audio=false&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&
↪jobmediafile=115&frame=467&alertid=1383&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`

13.14 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

13.14.1 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

13.15 CloudConvert Integration

New in version 4.6.

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

---

13.15.1 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
```

13.15.2 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>
```

13.15.3 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>
```

13.15.4 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.

13.15.5 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*

CHAPTER

FOURTEEN

TROUBLESHOOTING AND OBTAINING INFORMATION

14.1 Self test

New in version 4.0.

The Vidispine self test will generate a brief report about the system infrastructure for simple troubleshooting.

Note: (New in 4.2.) The Self test has moved from `APInoauth` to `API`.

14.1.1 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.

14.1.2 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.

14.1.3 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>
```

14.2 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 .

14.2.1 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.

14.2.2 Programmatically retrieving log files

New in version 3.3.

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

FIFTEEN

STANDALONE VIDISPINE

New in version 4.3.

The standalone Vidispine server is an application that can be used to run Vidispine without the need of an application server such as GlassFish or JBoss. The application embeds Jetty and OpenEJB, which makes it possible to run Vidispine in an application server or as a standalone service.

The goal is to:

- Make installing, configuring and upgrading VS easier.
- Give us full control of the libraries used with VS.

The differences compared to running Vidispine on GlassFish:

- Solr will NOT run in the same JVM as Vidispine.
- ActiveMQ is used instead of the broker (OpenMQ/HornetQ) used by the application server. ActiveMQ can be run embedded or standalone.

Here's what you need to know:

15.1 Installing distribution-specific packages

Use our packages for your distribution to install Vidispine.

15.1.1 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 8 which may not be available in the official repositories of older distributions. Install an thirdparty repository providing Java 8 or upgrade to a newer distribution.

Install from official repository

1. To install Vidispine directly from our repository, head over to the repository (http://repo.vidispine.com/) page and follow the instructions.

Install downloaded packages

1. Download the latest release from our download page (http://www.vidispine.com/partner/my-software).
2. Untar the downloaded package.

```
$ tar -xvzf vidispine-X.Y.tar.gz
$ cd vidispine-X-Y
```

3. Install the packages for your distribution:

   CentOS:

```
$ yum install vidispine-*el6*.rpm transcoder-*el6*.rpm
```

   Ubuntu:

```
$ dpkg -i vidispine-*.deb transcoder-*.deb
$ apt-get install -f
```

   This will install Vidispine, Java and any additional dependencies required by Vidispine. There are also optional dependencies that can be installed manually:

   - Graphviz - if you wish to *visualize users* or *jobs*.

15.1.2 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;
```

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
```

3. Initialize and migrate the database:

```
$ vidispine db ping # verify connection
$ vidispine db check # verify if tables exists (they shouldn't)
$ vidispine db init
$ 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.

15.1.3 Start the services

1. If you're on CentOS 6 or Ubuntu:

```
$ /etc/init.d/solr start
$ /etc/init.d/transcoder start
$ /etc/init.d/vidispine start
```

On systems using systemd:

```
$ 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
```

15.1.4 Configure Vidispine

Finally, you will need to *configure Vidispine*.

15.2 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
```

15.3 Service configuration

15.3.1 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.

15.3.2 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 is started before Vidispine. Note that this is only possible on systems running systemd.

```
$ vi /etc/systemd/system/vidispine.service.d/local.conf
[Unit]
Wants=transcoder.service solr.service

$ systemctl daemon-reload
$ systemctl enable vidispine
```

15.3.3 Setting JVM options

On Debian, edit the `/etc/default/vidispine` file:

```
$ vi /etc/default/vidispine
JAVA_OPTS="-Xmx8192m -XX:MaxPermSize=512m"
```

Or on CentOS 6, the file `/etc/sysconfig/vidispine`:

```
$ vi /etc/sysconfig/vidispine
JAVA_OPTS="-Xmx8192m -XX:MaxPermSize=512m"
```

On systems running systemd, create a file such as `/etc/systemd/system/vidispine.service.d/local.conf` and override the default JAVA_OPTS:

```
$ vi /etc/systemd/system/vidispine.service.d/local.conf
[Service]
Environment="JAVA_OPTS=-Xmx8192m -XX:MaxPermSize=512m"
```

15.4 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)
```

15.4.1 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]
```

15.5 Upgrading

15.5.1 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. On CentOS 6 or Ubuntu:

```
$ /etc/init.d/vidispine stop
```

On systems using systemd:

```
$ 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.

15.5.2 Upgrading from GlassFish

To move from GlassFish to the standalone server application:

1. Make sure that GlassFish is stopped.
2. Install the server and transcoder packages as described in the *installation guide*.
3. Modify the server configuration to match the GlassFish configuration.
   - Use the same database settings.
   - Add any custom JVM settings to the *service configuration*.
4. Migrate the database.
5. Start the `vidispine`, `solr` and `transcoder` services.

Once all steps have been performed and Vidispine is up and running:

1. Re-index items, collections and files to populate Solr.
2. Remove GlassFish.

Upgrading without re-indexing

It is also possible to avoid the re-index step by moving the existing Solr index to the new location used by the `solr` service.

1. Make sure that no application is using Vidispine.
2. Disable all services except the indexing service to make sure that all pending index updates are sent to Solr and removed from the JMS queue.

```
$ http put "localhost:8080/API/vidispine-service/disable"
$ http put "localhost:8080/API/vidispine-service/service/IndexCruncher/enable"
```

Then wait until all messages on the JMS queue "IndexQueue" have been processed by watching:

```
$ /opt/glassfish3/mq/bin/imqcmd list dst
```

3. Stop GlassFish.
4. Make sure that the vidispine-solr package is installed and that the `solr` service is *not* running.

```
$ /etc/init.d/solr/stop
```

```
$ systemctl stop solr
```

5. Move the Solr index to the new location.

```
$ mv /opt/glassfish3/glassfish/solrhome/solr/collection1/data/ /var/lib/vidispine/
↪solr/collection1/data
```

6. Start Solr and verify that the new index is in used by checking the document count from the Solr admin page at http://localhost:8983/solr/#/collection1.

15.6 Server configuration

Dropwizard 1.0 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 4.7: The Dropwizard version was updated from 0.7.1 to 1.0.

15.6.1 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}
```

15.6.2 Additional settings

In addition, the following properties are supported:

database

New in version 4.6.

- type: If this is a master/read-write or read-only database. Either `master` (default), `read_only_replica`, `read_only_snapshot`.

api

New in version 4.6.

- requestLogging: If *request logging* is enabled. Default is `true`.
- searchHistory: If *search history* is updated with new searches. Default is `true`.

secrets

New in version 4.7.

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.

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.

15.7 Package reference

15.7.1 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.

15.7.2 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.

15.7.3 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.

New in version 4.5: The lib/ext directory was added.

See also:

See our knowledge base articles (http://vidispine.tenderapp.com/kb/guides) for how to install Vidispine onto an application server.

CHAPTER

SIXTEEN

API REFERENCE

16.1 Access controls

16.1.1 Managing access controls

In the following reference, `{access-entity}` is one of the following:

- `/item`
- `/collection`
- `/library`

Retrieve access control list for an item

GET {access-entity}/ (*item-id*) **/access/**
Retrieves the entire access control list for the specified item.

Produces

- **application/xml, application/json** – AccessControlListDocument
- **text/plain** – CRLF-delimited list of ids

Role _accesscontrol_read

Add a new entry access control entry

POST {access-entity}/ (*item-id*) **/access/**
Adds a new access control entry for the specified item.

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 a specific access control entry

GET {access-entity}/(*item-id*)**/access/**
access-id Retrieves the desired access control entry.

Status Codes

- **404 Not found** – No entry with that id exists in that item.

Produces

- **application/xml, application/json** – An AccessControlDocument containing the requested access control entry.
- **text/plain** – The id of the entry.

Role _accesscontrol_read

Delete a specific access control entry

DELETE {access-entity}/(*item-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 item.

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

Remove all access control entries from all items

DELETE /item/access/
Deletes all access control entries from all known items.

Role _administrator

16.1.2 Managing access controls in bulk

Add multiple new entry access control entries

POST {access-entity}/ (*item-id*) **/access/bulk**
Adds multiple new access control entries to the specified item.

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>
    <recursive>true</recursive>
    <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>
    <recursive>true</recursive>
    <permission>READ</permission>
    <user>testUser</user>
  <access>
</AccessControlListDocument>
```

Delete multiple access control entries

DELETE {access-entity}/(*item-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
```

16.1.3 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** – 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
```

16.1.4 Viewing applied access controls

To review all access control entries that affects an item an AccessControlMergedDocument can be retrieved.

Retrieve a list of applied access control entries

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 entries will be listed according to priority for every user. If the access is given through a group or a collection, the names and ids of those will be given.

GET {access-entity}/ (*item-id*) **/merged-access/**

Query Parameters

- **username** – The name of the user to check.
- **permission** – The lowest required permission level.
- **type** – The type of operation to check for.
- **extradata** – Any possible extradata.

Produces

- **application/xml, application/json** – An AccessControlMergedDocument containing all access control that affects the item.

Role _accesscontrol_read

Example: retrieving all entries

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

```
<AccessControlMergedDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <access priority="1" id="VX-3111" username="admin">
    <permission>ALL</permission>
    <type>GENERIC</type>
  </access>
  <access priority="2" id="VX-24112" username="admin">
    <permission>WRITE</permission>
    <type>GENERIC</type>
    <collection>VX-10</collection>
  </access>
  <access priority="3" id="VX-4119" username="admin">
    <permission>ALL</permission>
    <type>GENERIC</type>
    <collection>VX-23</collection>
  </access>
  <access priority="4" id="VX-2221" username="admin">
    <permission>ALL</permission>
    <type>GENERIC</type>
    <collection>VX-12</collection>
  </access>
  <access priority="5" id="VX-2205" username="admin">
    <permission>ALL</permission>
    <type>GENERIC</type>
    <collection>VX-10</collection>
  </access>
  <access priority="1" id="VX-24090" username="test">
    <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-250. Notice that the access provided by VX-24112 does not match, but it is less prioritized than the access of VX-3111 and thus the user has full access to the

metadata.

```
GET /item/VX-250/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-250</item>
  </query>
  <access priority="1" matches="true" id="VX-3111">
    <permission>ALL</permission>
    <type>GENERIC</type>
  </access>
  <access priority="2" matches="false" id="VX-24112">
    <permission>WRITE</permission>
    <type>GENERIC</type>
    <collection>VX-10</collection>
  </access>
  <access priority="3" matches="true" id="VX-4119">
    <permission>ALL</permission>
    <type>GENERIC</type>
    <collection>VX-23</collection>
  </access>
  <access priority="4" matches="true" id="VX-2221">
    <permission>ALL</permission>
    <type>GENERIC</type>
    <collection>VX-12</collection>
  </access>
  <access priority="5" matches="true" id="VX-2205">
    <permission>ALL</permission>
    <type>GENERIC</type>
    <collection>VX-10</collection>
  </access>
</AccessControlMergedDocument>
```

Retrieve a list of applied access control entries that affects groups

GET {access-entity}/(*item-id*)**/merged-access/group**

Lists groups that have access to an item.

Even though a user belongs to a group that has access to an item, 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** –
  - `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.

New in version 4.2.3.

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>
```

16.1.5 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.

Get access graph

GET {access-entity}/(*item-id*)**/access/graph**
Shows the entity and any ancestor collections or libraries and the access controls on each.

Query Parameters

- **type** –
  - `ancestor` - Show the entity and ancestor entities in a hierarchy.
  - `grant` - Show users and how permissions have been granted.
- **users** – If the user and group hierarchy should be shown as a subgraph for the relevant users/groups.
- **groups** – If groups should be shown as nodes in the grant graph.

Produces

- **image/png** –

Role _administrator

Get access graph as dot file

GET {access-entity}/(*item-id*)**/access/graph/dot**

Shows the entity and any ancestor collections or libraries and the access controls on each.

Query Parameters

- **type** –
  - `ancestor` - Show the entity and ancestor entities in a hierarchy.
  - `grant` - Show users and how permissions have been granted.
- **users** – If the user and group hierarchy should be shown as a subgraph for the relevant users/groups.
- **groups** – If groups should be shown as nodes in the grant graph.

Produces

- **text/plain, text/vnd.graphviz** –

Role _administrator

16.2 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.

16.2.1 Examining the log

Retrieving log content

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`.

Query Parameters

- **path** – Optional string, matches path in log lines, default is `/`.

- **first** – Optional integer, number of first row to return, default is `0`.
- **rows** – Optional integer, number of rows to return, default is `100`. Cannot be greater than 1000.
- **starttime** – Optional ISO 8601 time, for lower limit of rows to return.
- **endtime** – Optional ISO 8601 time, for upper limit of rows to return.
- **wildcard** –
  - `false` - Do not treat do truncation at end of path.
  - `true` - Treat end of path to have a `*` wildcard.
- **username** – Optional string, only return rows that the specified user invoked. Default is all rows.
- **method** – Optional string, only return rows with the specified method, e.g. `GET`. Default is all rows.
- **performCount** –
  - `false` (default) - Do not return a total number of rows matching criteria.
  - `true` - Return a total number of rows matching criteria (except first and count).
- **sort** –
  - `asc` - Order by timestamp ascending.
  - `desc` (default) - Order by timestamp descending.

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>
```

Exporting log content

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** – Optional string, matches path in log lines, default is `/`.
- **first** – Optional integer, number of first row to return, default is `0`.
- **rows** – Optional integer, number of rows to return, default is `100`.
- **starttime** – Optional ISO 8601 time, for lower limit of rows to return.
- **endtime** – Optional ISO 8601 time, for upper limit of rows to return.
- **wildcard** –
  - `false` - Do not treat do truncation at end of path.
  - `true` - Treat end of path to have a `*` wildcard.
- **username** – Optional string, only return rows that the specified user invoked. Default is all rows.
- **method** – Optional string, only return rows with the specified method, e.g. `GET`. Default is all rows.
- **sort** –
  - `asc` - Order by timestamp ascending.
  - `desc` (default) - Order by timestamp descending.

Produces

- **application/xml** – AuditLogDocument

Role _administrator

16.3 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*.

16.3.1 Managing collections

Retrieve a list of all collections

GET /collection
Retrieves a list of all known collections.

Matrix Parameters

- **first** – Integer, from resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** – Integer, set a limit on maximum number of hits. Default `100`.

Query Parameters

- **count** –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **content** – Optional comma-separated list of addition content to retrieve. Possible values are: `metadata`, `merged-access`.
- **interval** – 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`
- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.
  - *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** – 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** – 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** – 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** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **include** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **defaultValue** –
  - `true` - For unset fields, return default values. See *Default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** –
  - `true` (default) - Include transient metadata, see *Transient metadata*.
  - `false` - Do not include transient metadata in response.
- **includeValues** – Return the value enumeration for each metadata field.
- **terse** –
  - `yes` - Return metadata in terse format (see *Retrieve terse metadata schema*)
  - `no` (default) - Return metadata in verbose format
- **revision** – Optional revision, specifying which metadata to display. Only used if requesting a single item or collection.
- **mergedType** – The type of operation to check for.
- **mergedPermission** – The lowest required permission level.
- **mergedExtradata** – Any possible extradata.

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** – Optional name of the collection.
- **externalId** – An external identifier to assign to the collection.
- **settings** – Pre-configured import settings. See *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

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.
- **404 Not found** – Could not find the collection.

Role _collection_write

Rename a collection

PUT /collection/(*collection-id*)**/rename**
Renames the collection with the *Identifiers* `collection-id`.

Query Parameters

- **name** – Mandatory, new name of the collection.

Role _collection_write

16.3.2 Collection content

Retrieve the contents of 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** – Optional comma-separated list of addition content to retrieve. Possible values are: `metadata`, `merged-access`.
- **interval** – 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`
- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.
  - *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** – 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** – 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** – 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** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **include** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.

- **defaultValue** –
  - `true` - For unset fields, return default values. See *Default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** –
  - `true` (default) - Include transient metadata, see *Transient metadata*.
  - `false` - Do not include transient metadata in response.
- **includeValues** – Return the value enumeration for each metadata field.
- **terse** –
  - `yes` - Return metadata in terse format (see *Retrieve terse metadata schema*)
  - `no` (default) - Return metadata in verbose format
- **revision** – Optional revision, specifying which metadata to display. Only used if requesting a single item or collection.
- **mergedType** – The type of operation to check for.
- **mergedPermission** – The lowest required permission level.
- **mergedExtradata** – Any possible extradata.

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.

Changed in version 4.4: 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** –
  - `list` (default) - Return a list of items.
  - `library` - Create a library with the matching items.
- **q** – XML/JSON, ItemSearchDocument. Only with GET (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3).
- **count** –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.

- **p** – Comma separated list of paths specifying the content to include. Overrides the content and filter parameters. See *Content paths*.

  New in version 4.5.

- **content** – Comma-separated list of the types of content to retrieve, possible values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.

- **interval** – 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`

- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.

  - *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** – 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** – 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** – 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** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.

- **includeValues** – Return the value enumeration for each metadata field.
- **conflict** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **terse** –
  - `yes` - Return metadata in terse format (see *Retrieve terse metadata schema*)
  - `no` (default) - Return metadata in verbose format
- **defaultValue** –
  - `true` - For unset fields, return default values. See *Default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** –
  - `true` (default) - Include transient metadata, see *Transient metadata*.
  - `false` - Do not include transient metadata in response.
- **revision** – Optional revision, specifying which metadata to display. Only used if requesting a single item or collection.
- **mergedType** – The type of operation to check for.
- **mergedPermission** – The lowest required permission level.
- **mergedExtradata** – Any possible extradata.
- **uriType** – Optional comma-separated list of format types (container format) to return.
- **scheme** – URI scheme to return.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodType** – Optional type of access method. See *Storages*.
  - `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** – Optional metadata used with storage method. See *Storages*.
- **tag** – A URI parameter, see *Get item URI*.
- **version** – Optional integer, 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** – A URI parameter, see *Get item URI*.
- **url** –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.
- **noauth-url** –
  - `true` Return URIs that do not need authentication.

- `false` (default) Return normal URIs

- **baseURI** – Which base URI to use for the thumbnail URLs.
- **save** –
  - `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

  New in version 4.7.

Matrix Parameters

- **library** – Restricts search to within library, *Identifiers*. Default is `*`, all items.
- **first** – Integer, from resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** – Integer, set a limit on maximum number of hits. Default `100`.
- **libraryId** – If set, the library identified by this id will be used instead of creating a new library.
- **autoRefresh** – See *Self-refreshing libraries*. Defaults to `false`.
- **updateMode** – See *Self-refreshing libraries*. Defaults to `MERGE`.
- **updateFrequency** – See *Self-refreshing libraries*. Defaults to no periodic updates.
- **starttc** –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.

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

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

New in version 4.0.

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*.

Changed in version 4.4: 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** –
  - `list` (default) - Return a list of items.
  - `library` - Create a library with the matching items.
- **q** – XML/JSON, ItemSearchDocument. Only with GET (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3).
- **count** –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **p** – Comma separated list of paths specifying the content to include. Overrides the content and filter parameters. See *Content paths*.

  New in version 4.5.
- **content** – Comma-separated list of the types of content to retrieve, possible values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** – 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`
- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.
  - *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** – 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** – 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** – 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** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **includeValues** – Return the value enumeration for each metadata field.
- **conflict** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **terse** –
  - `yes` - Return metadata in terse format (see *Retrieve terse metadata schema*)
  - `no` (default) - Return metadata in verbose format
- **defaultValue** –
  - `true` - For unset fields, return default values. See *Default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** –
  - `true` (default) - Include transient metadata, see *Transient metadata*.
  - `false` - Do not include transient metadata in response.

- **revision** – Optional revision, specifying which metadata to display. Only used if requesting a single item or collection.
- **mergedType** – The type of operation to check for.
- **mergedPermission** – The lowest required permission level.
- **mergedExtradata** – Any possible extradata.
- **uriType** – Optional comma-separated list of format types (container format) to return.
- **scheme** – URI scheme to return.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodType** – Optional type of access method. See *Storages*.
  - `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** – Optional metadata used with storage method. See *Storages*.
- **tag** – A URI parameter, see *Get item URI*.
- **version** – Optional integer, 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** – A URI parameter, see *Get item URI*.
- **url** –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.
- **noauth-url** –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **baseURI** – Which base URI to use for the thumbnail URLs.
- **save** –
  - `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

  New in version 4.7.

Matrix Parameters

- **library** – Restricts search to within library, *Identifiers*. Default is `*`, all items.
- **first** – Integer, from resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** – Integer, set a limit on maximum number of hits. Default `100`.
- **libraryId** – If set, the library identified by this id will be used instead of creating a new library.

- **autoRefresh** – See *Self-refreshing libraries*. Defaults to `false`.
- **updateMode** – See *Self-refreshing libraries*. Defaults to `MERGE`.
- **updateFrequency** – See *Self-refreshing libraries*. Defaults to no periodic updates.
- **starttc** –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.

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

Example

```
GET /collection/VX-76/item
Accept: application/xml
```

```
<ItemListDocument>
  <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** –
  - `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.
- **addItems** –
  - `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** –
  - *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`).

  New in version 4.7.

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=admin&metadata=dateAdded=Feb
```

```
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** –
  - `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.

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

New in version 4.7.

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** –
  - `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>
```

16.3.3 Collection metadata

Metadata can be set on collections, just like with items.

See *Metadata*.

16.3.4 Searching for collections

Searching collections behaves much like *Search*.

Search for collections

PUT /collection
Searches for collections that matches the query.

Matrix Parameters

- **first** – Integer. The index of the first collection. Default is `1`.
- **number** – Integer. The number of collections to retrieve. Default is `100`.

Query Parameters

- **count** –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **content** – Optional comma-separated list of addition content to retrieve. Possible values are: `metadata`, `merged-access`.
- **interval** – 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`
- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.
  - *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** – 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`** – 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`** – 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`** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **`include`** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **`defaultValue`** –
  - `true` - For unset fields, return default values. See *Default values*.
  - `false` (default) - Do not return default values.
- **`includeTransientMetadata`** –
  - `true` (default) - Include transient metadata, see *Transient metadata*.
  - `false` - Do not include transient metadata in response.
- **`includeValues`** – Return the value enumeration for each metadata field.
- **`terse`** –
  - `yes` - Return metadata in terse format (see *Retrieve terse metadata schema*)
  - `no` (default) - Return metadata in verbose format
- **`revision`** – Optional revision, specifying which metadata to display. Only used if requesting a single item or collection.
- **`mergedType`** – The type of operation to check for.
- **`mergedPermission`** – The lowest required permission level.
- **`mergedExtradata`** – Any possible extradata.

Accepts

- **`application/xml, application/json`** – ItemSearchDocument

Produces

- **`application/xml, application/json`** – CollectionListDocument

Role _collection_read

Retrieve search history

New in version 4.0.3.

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** – Optional. If set, only searches made after this date will be retrieved.
- **maxResults** – The maximum number of searches that will be retrieved. The value must be between 1 and 50, default is 10.
- **username** – 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

16.3.5 Ordering collections

Collections will return their elements in the same order for every request.

Changed in version 4.4: 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.

Reordering 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>
```

16.3.6 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

Un-mark 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** – The new name of the folder (required).

Produces

- **text/plain** – "OK"

Role _collection_write

16.4 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.

16.4.1 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

16.4.2 Indexing settings

Get 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

16.4.3 Metrics settings

See *Monitoring* for examples.

Get 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

16.4.4 Path alias configuration

See *Content paths* for information on paths and aliases.

Get 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
```

16.4.5 Job pool configuration

Get the job pool configuration

GET /configuration/job-pool
Returns the current job pool configuration.

New in version 4.2.2.

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.

New in version 4.2.2.

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.

New in version 4.2.2.

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 specific job pool

DELETE /configuration/job-pool/ (*priority*)
Deletes the job pool with the given priority threshold.

New in version 4.2.2.

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>
```

16.4.6 FTP pool configuration

Get the FTP pool configuration

GET /configuration/ftp-pool
Returns the current FTP connection pool configuration.

New in version 4.2.4.

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 job pool configuration

PUT /configuration/ftp-pool
Updates the FTP connection pool configuration.

New in version 4.2.4.

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.

New in version 4.2.4.

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"/>
```

16.4.7 LogReport configuration

Get the LogReport configuration

GET /configuration/logreport
Returns the current LogReport configuration.

Produces

- **application/xml, application/json** – LogReportConfigurationDocument

Role _administrator

Update the LogReport 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

16.4.8 Configuration properties

Get list of 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>
```

Get a single 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/modify 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/modify 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/
```

Remove 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
```

16.5 Export locations

New in version 4.0.

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.

16.5.1 Managing export locations

Listing available export locations

GET /export-location
List all defined export locations.

Produces

- **application/xml, application/json** – ExportLocationListDocument

Role _export

Creating/updating 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>
```

Get information about 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

Deleting an export location

DELETE /export-location/(*location-name*)
Delete the export location with the specified name.

Role _export

16.5.2 Export location script

Get 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

16.6 External identifiers

16.6.1 Managing external id 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>
```

Create or modify 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
```

16.6.2 Managing 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` (New in 4.5.)
Library	`/library/{library-id}/external-id` (New in 4.5.)
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

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

New in version 4.6.1.

DELETE /external-id/id/ (*external-id*)
Remove the external identifier from all entities.

Role _external_id_write

DELETE {external-id-resource}/ (*external-id*)
Remove 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
```

16.7 Groups and roles

16.7.1 Managing groups

List groups/roles

GET /group
Returns list of all groups.

Query Parameters

- **first** – Start returning groups from specified number. Default is `1`, the beginning of the list.
- **number** – Return at most specified number of groups. Default is no limit.

Produces

- **application/xml, application/json** – GroupListDocument
- **text/plain** – CRLF-delimited list of group names

Role _group_read

Get 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

Get 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 new 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

Create or update 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** –
  - `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

Search groups

PUT /group
Simple search of fields `groupname`, `description` and metadata.

Query Parameters

- **number** – Return at most specified number of groups. Default is `10`.

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>
```

16.7.2 Group information

Get group description

GET /group/ (*group-name*) **/description**
Returns the descriptive text about the specified group.

Produces

- **text/plain** – Group description

Role _group_read

Change 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

16.7.3 Group-to-group relations

Get parent groups to a group

GET /group/ (*group-name*) **/parents**
Returns groups that the specified group belongs to.

Query Parameters

- **traverse** –
  - `true` - Return all ancestors.
  - `false` (default) - Return only direct parents.

Produces

- **application/xml, application/json** – GroupListDocument
- **text/plain** – CRLF-delimeted list of *Tabbed tuples* of group name, group description

Role _group_read

Get child groups to a group

GET /group/ (*group-name*) **/children**
Returns groups that belongs to the specified group.

Query Parameters

- **traverse** –
  - `true` - Return all descendants.
  - `false` (default) - Return only direct children.

Produces

- **application/xml, application/json** – GroupListDocument
- **text/plain** – CRLF-delimeted list of *Tabbed tuples* of group name, group description

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

16.7.4 Group-to-user relations

Users belonging to group

GET /group/(*group-name*)**/users**
Returns all users belonging to the group/role, directly or indirectly.

Query Parameters

- **traverse** –
  - `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

16.8 Imports

16.8.1 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. For more information about jobs, see *Jobs*. Note that thumbnails and poster frames are only generated if a transcode takes place.

Query Parameters

- **uri** – A URI to the file that will be imported. See also *URI's, URL's, and Special Characters*.
- **URL** – A URL to the file that will be imported. (Deprecated since x.)
- **tag** – An optional list of *shape tags* to use for transcoding.
- **original** – An optional tag, if specified it 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** –
  - `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** –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **thumbnailService** – An optional identifier to which thumbnail resource that should be used.
- **createPosters** – An optional list of *time codes* to use for creating posters.
- **overrideFastStart** –
  - `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** –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** – Optional estimated duration of the clip in seconds.
- **storageId** – Optional identifier of storage where essence file is to be stored.
- **growing** –
  - `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** – An optional URI to a sidecar XMP metadata file.
- **sidecar** – Optional URIs or file ids of any sidecar files to import to the item. (New in 4.0.)
- **settings** – Pre-configured import settings. See *Import settings*
- **notification** – See *Notifications* . (Optional)

- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

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** – An optional list of *shape tags* to use for transcoding.
- **original** – An optional tag, if specified it 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** –
  - `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** –

  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **`thumbnailService`** – An optional identifier to which thumbnail resource that should be used.
- **`createPosters`** – An optional list of *time codes* to use for creating posters.
- **`transferId`** – An id to assign the transfer to be able to refer to it. Mandatory for chunked and passkey imports.
- **`transferPriority`** – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers (optional). A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **`storageId`** – Optional identifier of storage where essence file is to be stored.
- **`overrideFastStart`** –
  - `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`** –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **`fastStartLength`** – Optional estimated duration of the clip in seconds.
- **`notification`** – See *Notifications* . (Optional)
- **`notificationData`** – See *Notifications* . (Optional)
- **`priority`** – The priority to assign to the job. Default is `MEDIUM` .
- **`jobmetadata`** – Additional information for the job task. See *Special job metadata values*
- **`filename`** – The filename to be stored as original filename. Optional.
- **`settings`** – Pre-configured import settings. See *Import settings*
- **`ids`** – A comma-delimited list of external ids to assign to the item.

Status Codes

- **`400`** – If the amount of data received does not match the given Content-Length header.

  New in version 4.2.3.

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** – An optional list of *shape tags* to use for transcoding.
- **original** – An optional tag, if specified it 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** –
  - `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** –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **thumbnailService** – An optional identifier to which thumbnail resource that should be used.
- **createPosters** – An optional list of *time codes* to use for creating posters.
- **transferId** – An id to assign the transfer to be able to refer to it. Mandatory for chunked and passkey imports.
- **transferPriority** – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers (optional). A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **storageId** – Optional identifier of storage where essence file is to be stored.
- **overrideFastStart** –
  - `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** –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** – Optional estimated duration of the clip in seconds.
- **filename** – The filename to be stored as original filename. Optional.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*
- **settings** – Pre-configured import settings. See *Import settings*
- **ids** – A comma-delimited list of external ids to assign to the item.

Status Codes

- **400** – If the amount of data received does not match the given Content-Length header.

  New in version 4.2.3.

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** – An optional list of *shape tags* to use for transcoding.
- **original** – An optional tag, if specified it 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** –
  - `true` (default) - Generate thumbnails as per defined by shape tag.

  - `false` - Disable thumbnail generation.
- **`thumbnailService`** – An optional identifier to which thumbnail resource that should be used.
- **`createPosters`** – An optional list of *time codes* to use for creating posters.
- **`transferId`** – An id to assign the transfer to be able to refer to it. Mandatory for chunked and passkey imports.
- **`transferPriority`** – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers (optional). A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **`storageId`** – Optional identifier of storage where essence file is to be stored.
- **`overrideFastStart`** –
  - `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`** –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **`fastStartLength`** – Optional estimated duration of the clip in seconds.
- **`filename`** – The filename to be stored as original filename. Optional.
- **`notification`** – See *Notifications* . (Optional)
- **`notificationData`** – See *Notifications* . (Optional)
- **`priority`** – The priority to assign to the job. Default is `MEDIUM` .
- **`jobmetadata`** – Additional information for the job task. See *Special job metadata values*
- **`settings`** – Pre-configured import settings. See *Import settings*

Status Codes

- **`400`** – If the amount of data received does not match the given Content-Length header.

  New in version 4.2.3.

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>
```

16.8.2 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.
- **type** – Optional string.
  - `image-sequence`- Image sequence.
  - `dpx` - DPX sequence.

  Deprecated since version 4.6: Import image sequences using URIs with file fragments instead.
- **frameDuration** – Optional *Time durations* for each image in the sequence (optional).
- **settings** – Pre-configured import settings. See *Import settings*

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*)**/[container,audio,video]**

Imports the file and extracts component data based on what type is specified (container, audio, video). No transcoding will take place until all files have been imported.

Query Parameters

- **uri** – A URI to the file that will be imported. See also *URI's, URL's, and Special Characters*. Must be specified unless `fileId` is specified.
- **fileId** – The id of the file that contains the essence. Must be specified unless `uri` is specified.
- **allowReimport** –
  - `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** – An optional list of *shape tags* to use for transcoding.
- **original** – An optional tag, if specified it 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** –
  - `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** –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **thumbnailService** – An optional identifier to which thumbnail resource that should be used.
- **createPosters** – An optional list of *time codes* to use for creating posters.
- **storageId** – Optional identifier of storage where essence file is to be stored.
- **overrideFastStart** –
  - `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** –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** – Optional estimated duration of the clip in seconds.
- **growing** –
  - `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** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*
- **settings** – Pre-configured import settings. See *Import settings*
- **index** – The component index (track) of new component.

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*) **/[container,audio,video]/raw**

Imports the file and extracts component data based on what type is specified (container, audio, video). No transcoding will take place until all files have been imported.

Query Parameters

- **tag** – An optional list of *shape tags* to use for transcoding.
- **original** – An optional tag, if specified it 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** –
  - `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** –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **thumbnailService** – An optional identifier to which thumbnail resource that should be used.
- **createPosters** – An optional list of *time codes* to use for creating posters.
- **transferId** – An id to assign the transfer to be able to refer to it. Mandatory for chunked and passkey imports.
- **transferPriority** – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers (optional). A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **storageId** – Optional identifier of storage where essence file is to be stored.
- **overrideFastStart** –
  - `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** –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** – Optional estimated duration of the clip in seconds.
- **filename** – The filename to be stored as original filename. Optional.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Status Codes

- **400** – If the amount of data received does not match the given Content-Length header.

  New in version 4.2.3.

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). No transcoding will take place until all files have been imported.

Query Parameters

- **tag** – An optional list of *shape tags* to use for transcoding.
- **original** – An optional tag, if specified it 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** –
  - `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** –
  - `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** –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** – Optional estimated duration of the clip in seconds.
- **storageId** – Optional identifier of storage where essence file is to be stored.
- **growing** –
  - `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** – Pre-configured import settings. See *Import settings*
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*
- **index** – The starting component index (track) of the first component.

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.

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*) **/container/adopt/**

file-id Adopt the file as a container component in a placeholder item.

Role _import

16.8.3 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)
- 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.

Changed in version 4.7: Support for SCC subtitles was added.

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** – Either the file id of the sidecar file or a URL for locating it.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

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** – The id of the storage that the sidecar file will be saved.
- **fileExtension** – The extension of the file that this sidecar data is from. Used to identify the sidecar media type. Example: `srt`.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Status Codes

- **400** – If the amount of data received does not match the given Content-Length header.

  New in version 4.2.3.

Accepts

- **application/octet-stream** – The raw data.

Produces

- **application/xml, application/json** – A JobDocument that describes the import job.

Role _import

16.9 Import settings

16.9.1 Managing import settings

Create a new settings 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 the ids of all 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 a specific settings 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>
```

Change the settings of a 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 a profile

DELETE /import/settings/ (*settings-id*)
Deletes the profile with specified id.

Role _import

Example

```
DELETE /import/settings/VX-4
```

```
200 OK
```

16.10 Items

16.10.1 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.

Query Parameters

- **uri** – A URI to the destination of the file.
- **locationName** – (New in 4.0.) The name of an Export Location (see *Export locations*)
- **metadata** –
  - `true` - Metadata will also be exported to side-car XML file.
  - `false` (default) - No metadata is exported.
- **projection** – Defines the projection to use when exporting the metadata (optional).
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)

- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*
- **useOriginalFilename** – If set to true, the file(s) will be exported with their original filename if available.
- **tag** – 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** – Defines a start *Time codes* for the media (optional).
- **end** – Defines an end *Time codes* for the media (optional).
- **template** – Optional export template to use (see export-templates).
- **allowMissing** –
  - `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.

Produces

- **application/xml, application/json** – JobDocument

Role _export

Semantics

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.

Note: *FTP active mode*

New in version 4.1.2.

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*

New in version 4.1.4.

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 shape

New in version 4.1.2.

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** – A URI to the destination of the file.
- **locationName** – (New in 4.0.) The name of an Export Location (see *Export locations*)
- **metadata** –
  - `true` - Metadata will also be exported to side-car XML file.
  - `false` (default) - No metadata is exported.
- **projection** – Defines the projection to use when exporting the metadata (optional).
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*
- **useOriginalFilename** – If set to true, the file(s) will be exported with their original filename if available.
- **start** – Defines a start *Time codes* for the media (optional).
- **end** – Defines an end *Time codes* for the media (optional).
- **allowMissing** –
  - `true` (default) - Job will be started and the missing files will be ignored.
  - `false` - Job won't be started if there are missing files.

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** – A URI to the destination of the file.
- **locationName** – (New in 4.0.) The name of an Export Location (see *Export locations*)
- **metadata** –
  - `true` - Metadata will also be exported to side-car XML file.
  - `false` (default) - No metadata is exported.
- **projection** – Defines the projection to use when exporting the metadata (optional).
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*
- **useOriginalFilename** – If set to true, the file(s) will be exported with their original filename if available.
- **tag** – 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** –
  - `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** – Optional export template to use (see export-templates).

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>
```

16.10.2 Items

Managing items

Retrieve a list of 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** –
  - `list` (default) - Return a list of items.
  - `library` - Create a library with the matching items.
- **q** – XML/JSON, ItemSearchDocument. Only with GET (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3).
- **count** –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **p** – Comma separated list of paths specifying the content to include. Overrides the content and filter parameters. See *Content paths*.

  New in version 4.5.
- **content** – Comma-separated list of the types of content to retrieve, possible values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** – 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`
- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.
  - *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** – 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** – 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** – 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** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **includeValues** – Return the value enumeration for each metadata field.
- **conflict** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **terse** –
  - `yes` - Return metadata in terse format (see *Retrieve terse metadata schema*)
  - `no` (default) - Return metadata in verbose format
- **defaultValue** –
  - `true` - For unset fields, return default values. See *Default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** –
  - `true` (default) - Include transient metadata, see *Transient metadata*.
  - `false` - Do not include transient metadata in response.

- **revision** – Optional revision, specifying which metadata to display. Only used if requesting a single item or collection.
- **mergedType** – The type of operation to check for.
- **mergedPermission** – The lowest required permission level.
- **mergedExtradata** – Any possible extradata.
- **uriType** – Optional comma-separated list of format types (container format) to return.
- **scheme** – URI scheme to return.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodType** – Optional type of access method. See *Storages*.
  - `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** – Optional metadata used with storage method. See *Storages*.
- **tag** – A URI parameter, see *Get item URI*.
- **version** – Optional integer, 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** – A URI parameter, see *Get item URI*.
- **url** –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.
- **noauth-url** –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **baseURI** – Which base URI to use for the thumbnail URLs.
- **save** –
  - `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

  New in version 4.7.

Matrix Parameters

- **library** – Restricts search to within library, *Identifiers*. Default is `*`, all items.
- **first** – Integer, from resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** – Integer, set a limit on maximum number of hits. Default `100`.
- **libraryId** – If set, the library identified by this id will be used instead of creating a new library.

- **autoRefresh** – See *Self-refreshing libraries*. Defaults to `false`.
- **updateMode** – See *Self-refreshing libraries*. Defaults to `MERGE`.
- **updateFrequency** – See *Self-refreshing libraries*. Defaults to no periodic updates.
- **starttc** –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.

Produces

- **application/xml, application/json** – ItemListDocument
- **text/plain** – CRLF-delimited list of ids or URLs

Role _item_search

Tip: Additional content can be retrieved by using the syntax specified in *Retrieving item information*.

Get information about a single item

GET /item/ (*item-id*)
Returns information about a single item.

Query Parameters

- **p** – Comma separated list of paths specifying the content to include. Overrides the content and filter parameters. See *Content paths*.

  New in version 4.5.
- **content** – Comma-separated list of the types of content to retrieve, possible values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** – 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`
- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.

  - *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** – 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** – 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** – 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** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **includeValues** – Return the value enumeration for each metadata field.
- **conflict** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **terse** –
  - `yes` - Return metadata in terse format (see *Retrieve terse metadata schema*)
  - `no` (default) - Return metadata in verbose format
- **defaultValue** –
  - `true` - For unset fields, return default values. See *Default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** –
  - `true` (default) - Include transient metadata, see *Transient metadata*.
  - `false` - Do not include transient metadata in response.
- **revision** – Optional revision, specifying which metadata to display. Only used if requesting a single item or collection.
- **mergedType** – The type of operation to check for.
- **mergedPermission** – The lowest required permission level.
- **mergedExtradata** – Any possible extradata.

- **uriType** – Optional comma-separated list of format types (container format) to return.
- **scheme** – URI scheme to return.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodType** – Optional type of access method. See *Storages*.
  - `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** – Optional metadata used with storage method. See *Storages*.
- **tag** – A URI parameter, see *Get item URI*.
- **version** – Optional integer, 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** – A URI parameter, see *Get item URI*.
- **noauth-url** –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **baseURI** – Which base URI to use for the thumbnail URLs.

Matrix Parameters

- **starttc** –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.

Produces

- **application/xml, application/json** – ItemDocument

Tip: Additional content can be retrieved by using the syntax specified in *Retrieving item information*.

Delete a single item

Deleting an item means removing the item's id, its metadata, shapes, and physical files.

DELETE /item/(*item-id*)

Query Parameters

- **keepShapeTagMedia** – Optional comma-separated list of shape tags whose files will not be deleted.
- **keepShapeTagStorage** – Optional comma-separated list of storage ids whose files will not be deleted.

Semantics

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.

New in version 4.3.2.

If any of `keepShapeTagMedia` or `keepShapeTagStorage` contains a value `*`, then no files will be removed.

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`** –
  - `list` (default) - Return a list of items.
  - `library` - Create a library with the matching items.
- **`q`** – XML/JSON, ItemSearchDocument. Only with GET (http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.3).
- **`count`** –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **`p`** – Comma separated list of paths specifying the content to include. Overrides the content and filter parameters. See *Content paths*.

  New in version 4.5.
- **`content`** – Comma-separated list of the types of content to retrieve, possible values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **`interval`** – 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`
- **`field`** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.
  - *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** – 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** – 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** – 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** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **includeValues** – Return the value enumeration for each metadata field.
- **conflict** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **terse** –
  - `yes` - Return metadata in terse format (see *Retrieve terse metadata schema*)
  - `no` (default) - Return metadata in verbose format
- **defaultValue** –
  - `true` - For unset fields, return default values. See *Default values*.
  - `false` (default) - Do not return default values.

- **includeTransientMetadata** –
  - `true` (default) - Include transient metadata, see *Transient metadata*.
  - `false` - Do not include transient metadata in response.
- **revision** – Optional revision, specifying which metadata to display. Only used if requesting a single item or collection.
- **mergedType** – The type of operation to check for.
- **mergedPermission** – The lowest required permission level.
- **mergedExtradata** – Any possible extradata.
- **uriType** – Optional comma-separated list of format types (container format) to return.
- **scheme** – URI scheme to return.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodType** – Optional type of access method. See *Storages*.
  - `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** – Optional metadata used with storage method. See *Storages*.
- **tag** – A URI parameter, see *Get item URI*.
- **version** – Optional integer, 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** – A URI parameter, see *Get item URI*.
- **url** –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.
- **noauth-url** –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **baseURI** – Which base URI to use for the thumbnail URLs.
- **save** –
  - `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

  New in version 4.7.

Matrix Parameters

- **library** – Restricts search to within library, *Identifiers*. Default is `*`, all items.
- **first** – Integer, from resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.

- **number** – Integer, set a limit on maximum number of hits. Default `100`.
- **libraryId** – If set, the library identified by this id will be used instead of creating a new library.
- **autoRefresh** – See *Self-refreshing libraries*. Defaults to `false`.
- **updateMode** – See *Self-refreshing libraries*. Defaults to `MERGE`.
- **updateFrequency** – See *Self-refreshing libraries*. Defaults to no periodic updates.
- **starttc** –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.

Produces

- **application/xml, application/json** – ItemListDocument
- **text/plain** – CRLF-delimited list of ids or URLs

Accepts

- **application/xml, application/json** – ItemSearchDocument

Role _item_search

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** – Optional ISO8601 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** – 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

Creating 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** – The URI to output the CSV file to.
- **username** – Filter items according to the access of the specified user.
- **groupname** – Filter items according to the access of the specified group.
- **field** – A comma-separated list of metadata fields to include in the result.
- **outputFormat** – Specifies the output format. Valid values are `xml` (default) and `csv`.
- **data** – Specifies any additional data that should be included with the metadata fields.
- **p** – Comma-separated list of paths specifying the content to include. Overrides the field and data parameters. Only supported for XML output. See *Content paths*.

  New in version 4.5.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

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**

Produces

- **application/xml, application/json** – URIListDocument containing the collection ID of all collections that includes the item, and that the calling user has read access to.
- **text/plain** – CRLF-delimited list of ids

Role _item_read

Example

```
GET /item/VX-94/collections
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>VX-23</uri>
  <uri>VX-64</uri>
</URIListDocument>
```

16.10.3 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

- **p** – Comma separated list of paths specifying the content to include. Overrides the content and filter parameters. See *Content paths*.

  New in version 4.5.

- **content** – Comma-separated list of the types of content to retrieve, possible values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** – A metadata parameter, see *Get metadata*.
- **field** – A metadata parameter, see *Get metadata*.
- **group** – A metadata parameter, see *Get metadata*.
- **language** – A metadata parameter, see *Get metadata*.
- **sampleRate** – A metadata parameter, see *Get metadata*.
- **track** – A metadata parameter, see *Get metadata*.
- **terse** – A metadata parameter, see *Get metadata*.
- **include** – A metadata parameter, see *Get metadata*.
- **type** – A URI parameter, see *Get item URI*.
- **tag** – A URI parameter, see *Get item URI*.
- **scheme** – A URI parameter, see *Get item URI*.
- **closedFiles** – A URI parameter, see *Get item URI*.
- **noauth-url** – If `true`, thumbnail URIs that do not need authentication are returned. If `false` (default), normal thumbnail URIs are returned.
- **defaultValue** – A metadata parameter, see *Get metadata*.
- **methodType** – Optional type of storage method. When returning URIs, only use methods of this type. See *Storages*.
- **methodMetadata** – Optional metadata used with storage method. See *Storages*.
- **version** – Optional integer, 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.
- **revision** – Optional revision, specifying which metadata to display. Only used if requesting a single item or collection.

Produces

- **application/xml, application/json** – ItemDocument

Role _metadata_read

Role _thumbnail_read

Role _item_uri

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>
    <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** – Optional comma-separated list of format types (container format) to return.
- **tag** – Optional comma-separated list of shape tags to return. See *Shape tags*.
- **scheme** – Optional URI scheme to return, e.g. `ftp`.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodType** – Optional type of access method. See *Storages*.
  - `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** – Optional metadata used with storage method. See *Storages*.
- **closedFiles** –
  - `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>
```

16.10.4 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** – An ISO 8601 compatible timestamp (**optional**). Defaults to the current time.
- **duration** – An ISO 8601 compatible duration (**optional**). Defaults to zero.

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 information about 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>
```

Remove 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** – An ISO 8601 compatible timestamp. Defaults to the current time.
- **duration** – An ISO 8601 compatible duration. Defaults to zero.

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
```

16.10.5 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

Get list of 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** –
  - `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).

Generate a new 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** – Mandatory parameter.
  - `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

- **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.

Retrieve a specific 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** – Optional parameter.
  - `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

16.10.6 Item sequences

Sequence operations

List the available 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

Create/update a sequence

PUT /item/ (*id*) **/sequence/**

format Creates or updates the sequence in the given format.

Query Parameters

- **pauseFrame** – 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

Remove 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

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

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`** – The shape tag specifying the format of the rendered sequence.

- **sourceTag** – The shape tag specifying the shapes to use as input.
- **original** – An optional tag, if specified it 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** – An item id, to which the new new shape will be associated.

  New in version 4.4.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

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

Rendering a sequence on an item

New in version 4.1.1.

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** – The shape tag specifying the format of the rendered sequence.
- **sourceTag** – The shape tag specifying the shapes to use as input.
- **subtitleLanguage** – The language code specifying the subtitle language to use, see *Subtitles*.

  New in version 4.7.
- **original** – An optional tag, if specified it 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** – An item id, to which the new new shape will be associated.

  New in version 4.4.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

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>
```

16.10.7 Shapes

Item shapes

Get list of shapes

GET /item/(*id*)**/shape**

Returns all existing shapes for a specified item.

Query Parameters

- **version** –
  - *essence-version-id* - Return shapes for a specified version.
  - `all` - Return shapes for all versions.
  - `latest` (default) - Return shapes for the latest version.
- **tag** – Optional comma-separated list. Only return shapes with these tags.
- **url** –
  - `true` - Return list of URLs.
  - `false` (default) - Return list of ids.
- **placeholder** –
  - `true` - Only return placeholder shapes.
  - `false` (default) - Only return non-placeholder shapes.

    New in version 4.2.3.

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

Get shape

GET /item/(*id*)**/shape/**
shape-id Returns a shape for a specified item.

Query Parameters

- **methodType** – Return URIs from storage methods with a particular type. By default, return URLs with empty `methodType`. see *Storage methods*.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodMetadata** – Optional metadata used with storage method. See *Storages*.
- **scheme** – URI scheme to return.
- **transient** –
  - `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** –
  - `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

Deleting 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** –
  - `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** –

- `true` - Keep the files belong to this shape.
- `false` (default) - Remove the files belong to this shape.

- **updateMetadata** –
  - `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

New in version 4.7.

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 *Deleting an essence version of an item*.

Query Parameters

- **keepFiles** –
  - `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** – A URI to the file that will be imported. See also *URI's, URL's, and Special Characters*. Must be specified unless `fileId` is specified.
- **fileId** – The id of the file that contains the essence. Must be specified unless `uri` is specified.
- **allowReimport** –
  - `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** – The tags to assign to the new shape.
- **storageId** – Optional identifier of storage where essence file is to be stored.

  New in version 4.7.

- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

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** – The tags to assign to the new shape.
- **storageId** – Optional identifier of storage where essence file is to be stored.

  New in version 4.7.

- **filename** – The filename to be stored as original filename. Optional.
- **transferId** – An id to assign the transfer to be able to refer to it. Mandatory for chunked and passkey imports.
- **transferPriority** – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers (optional). A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Status Codes

- **400** – If the amount of data received does not match the given Content-Length header.

  New in version 4.2.3.

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.

Query Parameters

- **tag** – The tags to assign to the new shape.
- **updateItemMetadata** – If the shape is tagged `original` and this query parameter is true, the item's system metadata (e.g. `durationSeconds`) is updated.

Accepts

- **application/xml, application/json** – ShapeDocument

Produces

- **application/xml, application/json** – ShapeDocument

Role _import

Creating thumbnails and posters

Thumbnails and posters of a specific shape can be created by starting a thumbnail job.

Start a thumbnail job

New in version 4.2.3.

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.

Query Parameters

- **createThumbnails** –
  - `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** – An optional list of *Time codes* to use for creating posters.
- **thumbnailWidth** – An optional integer specifying the width of the thumbnails.
- **thumbnailHeight** – An optional integer specifying the height of the thumbnails.
- **thumbnailPeriod** – An optional 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** – An optional integer specifying the width of the posters.

- **posterHeight** – An optional integer specifying the height of the posters.
- **posterFormat** –
  - `jpeg` (default) - Creates posters in JPEG format.
  - `png` - Creates posters in PNG format.
- **thumbnailService** – An optional identifier to which thumbnail resource that should be used.
- **tag** – Include additional video settings from this transcode preset.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

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*.

Get all essence versions for an item

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 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** – A URI to the file that will be imported. See also *URI's, URL's, and Special Characters*. Must be specified unless `fileId` is specified.
- **fileId** – The id of the file that contains the essence. Must be specified unless `uri` is specified.
- **allowReimport** –
  - `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** – The tags to assign to the new shape.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Produces

- **application/xml, application/json** – A JobDocument that describes the import job.

Role _import

Import 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** – The tags to assign to the new shape.
- **storageId** – Optional identifier of storage where essence file is to be stored.
- **filename** – The filename to be stored as original filename. Optional.
- **transferId** – An id to assign the transfer to be able to refer to it. Mandatory for chunked and passkey imports.
- **transferPriority** – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers (optional). A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Status Codes

- **400** – If the amount of data received does not match the given Content-Length header.

  New in version 4.2.3.

Accepts

- **application/octet-stream** – The raw essence data.

Produces

- **application/xml, application/json** – A JobDocument that describes the import job.

Role _import

Get a particular essence versions for an item

GET /item/ (*id*) **/shape/version/**
version-number 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

Deleting an essence version of an item

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

Placeholder shapes

Create a placeholder shape

POST /item/ (*id*) **/shape/placeholder**
Creates an new placeholder shape for a specific item.

Query Parameters

- **tag** – Optional comma separated shape tags to be added to the shape.
- **container** – Optional integer, the number of container components
- **audio** – Optional integer, the number of audio components
- **video** – Optional integer, the number of video components
- **frameDuration** – Optional *Time durations* for each image in the sequence (optional).

Accepts

- **application/xml, application/json** – SimpleMetadataDocument

Produces

- **text/plain** – The id of the new shape.

Role _import

Modify a placeholder shape

PUT /item/ (*id*) **/shape/**
*shape-id***/placeholder** Updates the expected number of container, video and audio components for a specific placeholder shape.

Query Parameters

- **tag** – Optional comma separated shape tags to be added to the shape.
- **container** – Optional integer, the number of container components

- **audio** – Optional integer, the number of audio components
- **video** – Optional integer, the number of video components

Accepts

- **application/xml, application/json** – SimpleMetadataDocument

Role _import

Shape files

Get files for shape

GET /item/(*id*)**/shape/***shape-id*`/file` Returns all files that are associated with the specified shape.

Matrix Parameters

- **includeItem** –
  - `true` - Return associated items, shapes, and components.
  - `false` (default) - Do not return any information about associated items, shapes, and components.

Query Parameters

- **closedFiles** –
  - `true` (default) - Return only files that are closed.
  - `false` - Return all files.
- **methodType** – Return URIs from storage methods with a particular type. By default, return URLs with empty `methodType`. see *Storage methods*.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodMetadata** – Optional metadata used with storage method. See *Storages*.
- **scheme** – URI scheme to return.

Produces

- **application/xml, application/json** – FileListDocument

Role _item_shape_read

Shape metadata

New in version 4.0.

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** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Produces

- **application/xml, application/json** – JobDocument

Role _item_shape_write

Tags of a shape

Get a list of tags associated with a shape

GET /item/ (*item-id*) **/shape/**
*shape-id***/tag/** Retrieves all shape tags associated with a certain shape.

Query Parameters

- **url** –
  - `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

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

DELETE /item/ (*id*) **/shape/**
*shape-id***/mime/***mime-type* Removes a mime type from the shape.

Role _item_shape_write

16.10.8 Shape analysis

Shapes can be analyzed to detect for example detect cropping and silence.

New in version 4.4.

Analysis will also generation information for generating waveform data, to be visualized in a UI.

Operations

Analyze a shape

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 *RestBulkyMetadata* of the shape.

Query Parameters

- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Accepts

- **application/xml, application/json** – AnalyzeJobDocument

Produces

- **application/xml, application/json** – JobDocument

Role _job_write

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"/>
```

```
<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.

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

New in version 4.4.

The waveform data is not exactly the waveform, but measurements of the RMS values with a rather high sampling rate.

Get waveform data

New in version 4.5.1.

GET /item/ (*item-id*) **/waveform/values**

Query Parameters

- **itemTrack** – The itemTrack value of the audio channel within the shape. (Optional)
- **stream** – The stream value of the audio channel within the component of the shape. (Optional)
- **channel** – The channel value of the audio channel within the stream of the component. Optional, if omitted the first channel. 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.
- **shape** – The shape id to use to get information from. Optional, if omitted the shape tag will be used. Note that an analysis of this shape must be done before the information is available.
- **tag** – The shape tag to use. Optional, if omitted the `original` shape tag will be used.
- **start** – The start time code to get waveform information for. Default is `-INF`.
- **end** – The end time code to get waveform information for. Default is `+INF`.
- **dB** –

  true Return RMS dB values.

  false (default) Return RMS 1-based absolute values.
- **width** – 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**

Query Parameters

- **itemTrack** – The itemTrack value of the audio channel within the shape. (Optional)
- **stream** – The stream value of the audio channel within the component of the shape. (Optional)
- **channel** – The channel value of the audio channel within the stream of the component. Optional, if omitted the first channel. 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.
- **shape** – The shape id to use to get information from. Optional, if omitted the shape tag will be used. Note that an analysis of this shape must be done before the information is available.
- **tag** – The shape tag to use. Optional, if omitted the `original` shape tag will be used.
- **start** – The start time code to get waveform information for. Default is `-INF`.
- **end** – The end time code to get waveform information for. Default is `+INF`.
- **dB** –

  true Return RMS dB values.

  false (default) Return RMS 1-based absolute values.
- **width** – 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**

Query Parameters

- **itemTrack** – The itemTrack value of the audio channel within the shape. (Optional)
- **stream** – The stream value of the audio channel within the component of the shape. (Optional)
- **channel** – The channel value of the audio channel within the stream of the component. Optional, if omitted the first channel. 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.

- **shape** – The shape id to use to get information from. Optional, if omitted the shape tag will be used. Note that an analysis of this shape must be done before the information is available.
- **tag** – The shape tag to use. Optional, if omitted the `original` shape tag will be used.
- **start** – The start time code to get waveform information for. Default is `-INF`.
- **end** – The end time code to get waveform information for. Default is `+INF`.
- **dB** –

  true Return RMS dB values.

  false (default) Return RMS 1-based absolute values.
- **width** – The number of sample points to return. Default is 400.
- **height** – The height, in pixels, of the image. Default is 100.
- **bgcolor** – The background color of the image, as hex triplet. Defaults to `#000000` (black).
- **fgcolor** – The color of the waveform, as hex triplet. Defaults to white.
- **hgridline** – The position of primary horizontal gridlines, in units of the audio. Default is no gridline.
- **hgridlinecolor** – The color of primary horizontal gridlines.
- **hgridline2** – The position of secondary horizontal gridlines, in units of the audio. Default is no gridline.
- **hgridline2color** – The color of secondary horizontal gridlines.
- **vgridline** – The position of primary vertical gridlines, where 0 is left border and 1 is right border. Default is no gridline.
- **vgridlinecolor** – The color of primary vertical gridlines.
- **vgridline2** – The position of primary vertical gridlines, where 0 is left border and 1 is right border. Default is no gridline.
- **vgridline2color** – The color of primary vertical gridlines.
- **min** – The audio value that corresponds the bottom border. Defaults to -1 if `dB` is false, and -80 otherwise.
- **max** – The audio value that corresponds the top border. Defaults to 1 if `dB` is false, and 0 otherwise.

Produces

- **image/png** – A PNG image with the waveform drawn on the canvas described by the query parameters.

Role _item_shape_read

Get waveform image URI

New in version 4.5.1.

GET /item/ (*item-id*) **/waveform/imageURI**

Query Parameters

- **itemTrack** – The itemTrack value of the audio channel within the shape. (Optional)
- **stream** – The stream value of the audio channel within the component of the shape. (Optional)
- **channel** – The channel value of the audio channel within the stream of the component. Optional, if omitted the first channel. 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.
- **shape** – The shape id to use to get information from. Optional, if omitted the shape tag will be used. Note that an analysis of this shape must be done before the information is available.
- **tag** – The shape tag to use. Optional, if omitted the `original` shape tag will be used.
- **start** – The start time code to get waveform information for. Default is `-INF`.
- **end** – The end time code to get waveform information for. Default is `+INF`.
- **dB** –

  true Return RMS dB values.

  false (default) Return RMS 1-based absolute values.
- **width** – The number of sample points to return. Default is 400.
- **height** – The height, in pixels, of the image. Default is 100.
- **bgcolor** – The background color of the image, as hex triplet. Defaults to `#000000` (black).
- **fgcolor** – The color of the waveform, as hex triplet. Defaults to white.
- **hgridline** – The position of primary horizontal gridlines, in units of the audio. Default is no gridline.
- **hgridlinecolor** – The color of primary horizontal gridlines.
- **hgridline2** – The position of secondary horizontal gridlines, in units of the audio. Default is no gridline.
- **hgridline2color** – The color of secondary horizontal gridlines.
- **vgridline** – The position of primary vertical gridlines, where 0 is left border and 1 is right border. Default is no gridline.
- **vgridlinecolor** – The color of primary vertical gridlines.
- **vgridline2** – The position of primary vertical gridlines, where 0 is left border and 1 is right border. Default is no gridline.
- **vgridline2color** – The color of primary vertical gridlines.
- **min** – The audio value that corresponds the bottom border. Defaults to -1 if `dB` is false, and -80 otherwise.
- **max** – The audio value that corresponds the top border. Defaults to 1 if `dB` is false, and 0 otherwise.

Produces

- **text/plain** – A URI that does not require authentication to the generated image. The URI expires after 1 hour.

Role _item_shape_read

Get waveform images for all audio channels

Solely used for debugging. May be deprecated in newer releases.

GET /item/ (*item-id*) **/waveform/alltracks**

Query Parameters

- **itemTrack** – The itemTrack value of the audio channel within the shape. (Optional)
- **stream** – The stream value of the audio channel within the component of the shape. (Optional)
- **channel** – The channel value of the audio channel within the stream of the component. Optional, if omitted the first channel. 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.
- **shape** – The shape id to use to get information from. Optional, if omitted the shape tag will be used. Note that an analysis of this shape must be done before the information is available.
- **tag** – The shape tag to use. Optional, if omitted the `original` shape tag will be used.
- **start** – The start time code to get waveform information for. Default is `-INF`.
- **end** – The end time code to get waveform information for. Default is `+INF`.
- **dB** –

  true Return RMS dB values.

  false (default) Return RMS 1-based absolute values.
- **width** – The number of sample points to return. Default is 400.
- **height** – The height, in pixels, of the image. Default is 100.
- **bgcolor** – The background color of the image, as hex triplet. Defaults to `#000000` (black).
- **fgcolor** – The color of the waveform, as hex triplet. Defaults to white.
- **hgridline** – The position of primary horizontal gridlines, in units of the audio. Default is no gridline.
- **hgridlinecolor** – The color of primary horizontal gridlines.
- **hgridline2** – The position of secondary horizontal gridlines, in units of the audio. Default is no gridline.
- **hgridline2color** – The color of secondary horizontal gridlines.
- **vgridline** – The position of primary vertical gridlines, where 0 is left border and 1 is right border. Default is no gridline.
- **vgridlinecolor** – The color of primary vertical gridlines.
- **vgridline2** – The position of primary vertical gridlines, where 0 is left border and 1 is right border. Default is no gridline.
- **vgridline2color** – The color of primary vertical gridlines.
- **min** – The audio value that corresponds the bottom border. Defaults to -1 if `dB` is false, and -80 otherwise.

- **max** – The audio value that corresponds the top border. Defaults to 1 if `dB` is false, and 0 otherwise.

Produces

- **text/html** – An HTML document including image references to waveform images for all channels. Query parameters can be used to control the image appearance.

Role _item_shape_read

16.10.9 Shape components

Components

Get components for 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

Get component information

GET /item/ (*id*) **/shape/**
*shape-id***/component/***component-id* Returns all files for a specified component.

Produces

- **application/xml, application/json** – FileListDocument
- **text/plain** – List of file URLs

Role _item_shape_read

Component import

Import 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** – The URI to the file containing the new shape.
- **fileId** – The id of the file that contains the new shape.
- **allowReimport** –

- `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** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Produces

- **application/xml, application/json** – JobDocument

Status Codes

- **400** – If the file has already been imported.

Role _import

Move/copy components

Move 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** – The component index (track) of component. If the target shape has a component with this index, then it will be replaced/removed.
- **keepMetadata** –
  - `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 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** – The component index (track) of component.
- **keepMetadata** –
  - `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 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`** – The component index (track) of component. If the target shape has a component with this index, then it will be replaced/removed.
- **`keepMetadata`** –
  - `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 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`** – The component index (track) of component.
- **`keepMetadata`** –
  - `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

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** – The type of component. Either `audio`, `video` or `container`.
- **index** – The component index (track) of new component.

Produces

- **application/xml, application/json** – ComponentDocument

Component metadata

New in version 4.0.

Please refer to *Key-value metadata*.

16.10.10 Thumbnails

Creating thumbnails and posters

Thumbnails and posters can be created by starting a thumbnail job.

Start 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.

Query Parameters

- **createThumbnails** –
  - `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** – An optional list of *Time codes* to use for creating posters.
- **thumbnailWidth** – An optional integer specifying the width of the thumbnails.
- **thumbnailHeight** – An optional integer specifying the height of the thumbnails.
- **thumbnailPeriod** – An optional 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** – An optional integer specifying the width of the posters.
- **posterHeight** – An optional integer specifying the height of the posters.
- **posterFormat** –
  - `jpeg` (default) - Creates posters in JPEG format.
  - `png` - Creates posters in PNG format.
- **thumbnailService** – An optional identifier to which thumbnail resource that should be used.

- **tag** – Include additional video settings from this transcode preset.
- **version** – An optional version number. For creating thumbnails for older versions of the item essence. Default is latest version.
- **sourceTag** – Optional 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.

  New in version 4.2.3.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Produces

- **application/xml, application/json** – JobDocument

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.

Get 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** – 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

New in version 4.2.

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** – 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

Put resources 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

New in version 4.2.

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.

Thumbnail resource handling

The following requests deal with managing collections of thumbnail URIs for a specific thumbnail resource.

Get list of thumbnails

GET {thumbnail-resource}
Returns thumbnail URIs on which further requests may be performed.

Query Parameters

- **url** –
  - `true` - Return list of URLs.

- `false` (default) - Return list of ids.

- **noauth-url** –
  - `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

Insert or update 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

Remove 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.

Get image representation

GET {thumbnail-resource}/(*time*)
Return the image representation of this thumbnail.

Produces

- **image/png, image/jpeg** – Image of the thumbnail

Role _thumbnail_read

Remove thumbnail

DELETE {thumbnail-resource}/ (*time*)
Remove this thumbnail.

Role _thumbnail_write

Export thumbnail

POST {thumbnail-resource}/ (*time*) **/export**

Query Parameters

- **uri** – Mandatory URI of export location of thumbnail or poster
- **format** – Optional image format of destination. E.g. `tiff`.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Produces

- **application/xml, application/json** – JobDocument

Role _thumbnail_read

16.10.11 Transcoding

Transcoding

Start an item transcode job

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** – A comma-separated list of shape tags, that specify the desired output.
- **createThumbnails** –
  - `true` - Creates thumbnails according to default transcoder rules.
  - `false` (default) - No thumbnails will be created.
- **createPosters** – An optional list of *Time codes* to use for creating posters.
- **thumbnailService** – An optional identifier to which thumbnail resource that should be used.
- **destinationItem** – An item id, to which the new new shape will be associated.

  New in version 4.4.

- **original** – An optional tag, if specified it 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.

  New in version 4.6.

- **overrideFastStart** –
  - `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** –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** – Optional estimated duration of the clip in seconds.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Produces

- **application/xml, application/json** – JobDocument

Start a shape transcode job

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** – A comma-separated list of shape tags, that specify the desired output.
- **createThumbnails** –
  - `true` - Creates thumbnails according to default transcoder rules.
  - `false` (default) - No thumbnails will be created.
- **createPosters** – An optional list of *Time codes* to use for creating posters.
- **thumbnailService** – An optional identifier to which thumbnail resource that should be used.
- **destinationItem** – An item id, to which the new new shape will be associated.

  New in version 4.4.

- **original** – An optional tag, if specified it 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.

  New in version 4.6.

- **overrideFastStart** –
  - `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`** –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **`fastStartLength`** – Optional estimated duration of the clip in seconds.
- **`notification`** – See *Notifications* . (Optional)
- **`notificationData`** – See *Notifications* . (Optional)
- **`priority`** – The priority to assign to the job. Default is `MEDIUM` .
- **`jobmetadata`** – Additional information for the job task. See *Special job metadata values*

Produces

- **`application/xml, application/json`** – JobDocument

16.10.12 Item conform

New in version 4.0.

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`** –
  - `true` (default) - Copy metadata from the source items, according to the timeline, to the resulting item.
  - `false` - Do not copy metadat from the source items.

  New in version 4.2.3.
- **`sourceTag`** – A comma-separated list of shape tags, that specify the shapes that should be used as inputs.
- **`original`** – An optional tag, if specified it 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.
- **`tag`** – A comma-separated list of shape tags, that specify the desired output.
- **`createThumbnails`** –

- `true` - Creates thumbnails according to default transcoder rules.
- `false` (default) - No thumbnails will be created.

- **`createPosters`** – An optional list of *Time codes* to use for creating posters.
- **`thumbnailService`** – An optional identifier to which thumbnail resource that should be used.
- **`destinationItem`** – An item id, to which the new new shape will be associated.

  New in version 4.4.
- **`notification`** – See *Notifications* . (Optional)
- **`notificationData`** – See *Notifications* . (Optional)
- **`priority`** – The priority to assign to the job. Default is `MEDIUM` .
- **`jobmetadata`** – Additional information for the job task. See *Special job metadata values*

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`** –
  - `true` (default) - Copy metadata from the source items, according to the timeline, to the resulting item.
  - `false` - Do not copy metadat from the source items.

  New in version 4.2.3.
- **`sourceTag`** – A comma-separated list of shape tags, that specify the shapes that should be used as inputs.
- **`original`** – An optional tag, if specified it 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.
- **`tag`** – A comma-separated list of shape tags, that specify the desired output.
- **`createThumbnails`** –
  - `true` - Creates thumbnails according to default transcoder rules.
  - `false` (default) - No thumbnails will be created.
- **`createPosters`** – An optional list of *Time codes* to use for creating posters.

- **thumbnailService** – An optional identifier to which thumbnail resource that should be used.
- **destinationItem** – An item id, to which the new new shape will be associated.

  New in version 4.4.
- **overrideFastStart** –
  - `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** –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** – Optional estimated duration of the clip in seconds.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

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

<ConformRequest 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>
</ConformRequest>
```

```
<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>
```

16.10.13 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

Get List of Timeline Representations

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.

Get Timeline Representation

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

Create/Modify Timeline

PUT /item/(*id*)**/timeline/**
timeline-format Creates or updates 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

Remove All Timeline Representations

DELETE /item/(*id*)**/timeline**
Removes all timeline representations.

Status Codes

- **200** – All timelines associated with the item were deleted successfully.

Role _item_timeline_write

Remove Specific Timeline Representations

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

16.11 JavaScript

16.11.1 Testing scripts

New in version 4.0.3.

In order to test functionality, the JavaScript engine can be called manually.

Executing JavaScript code

`POST /javascript/test`

Accepts

- **`application/javascript , text/plain, text/javascript`** – The JavaScript code

Produces

- **`application/json`** – The object returned by the code, in JSON format

Role _administrator

16.11.2 JavaScript sessions

See *Debugging JavaScript* on how to debug JavaScript scripts.

Retrieve JavaScript sessions

`GET /javascript/session`

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 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** –
  - `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

16.12 Jobs

16.12.1 Managing jobs

Create job

See *Creating jobs*.

Get list of jobs

GET /job
Return jobs matching the criteria given.

Query Parameters

- **jobmetadata** – Optional comma-separated list
  - *key* = *value* - Filter out only the jobs that has job metadata according to the filter criteria.

  Note that = is part of the query parameter, and has to be encoded (`%3d`).
- **metadata** –
  - `true` - Include job metadata with all jobs.
  - `false` (default) - Do not include job metadata with all jobs.
- **idonly** –
  - `true` - Only return a list of ids
  - `false` (default) - Return job information such as job status
- **starttime-from** – Optional ISO8601 timestamp. Return only jobs started after the given timestamp.
- **starttime-to** – Optional ISO8601 timestamp. Return only jobs started before the given timestamp.
- **step** –
  - `true` - Include step information in the job listing.
  - `false` (default) - Do not include step information in the job listing.

Matrix Parameters

- **type** – Optional comma-separated list of *Job types*. Default is `all`, return all jobs.
- **state** – Optional comma-separated list of *Job states*. Default is `all`, return all jobs.
- **first** – Optional integer. Return jobs from that number in the list of sorted jobs. Default is `1`, the first jobs.
- **number** – Optional integer. Return at most that number of jobs. Default is all jobs.
- **sort** – Optional list of form `field (asc|desc)[,...]`. Sort by specific fields.
  - `jobId`
  - `type`
  - `state`
  - `user`
  - `startTime`
  - `priority`

  New in version 4.0.1.
- **user** –
  - `true` (default) - Include only jobs created by current user
  - `false` - Include all jobs

Produces

- **application/xml, application/json** – JobListDocument
- **text/plain** – CRLF-delimited list of job ids.

Role _job_read

Get job information

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** –
  - `true` - Include job metadata with all jobs.
  - `false` (default) - Do not include job metadata with all jobs.

Status Codes

- **404 Not found** – Invalid id

Produces

- **application/xml, application/json** – JobDocument
- **text/plain** – State of job

Role _job_read

Modify job parameter

PUT /job/ (*job-id*)
Updates the job by setting job priority

Query Parameters

- **priority** – Change the job priority. Value should be one of `LOWEST`, `LOW`, `MEDIUM`, `HIGH` or `HIGHEST`.

Status Codes

- **404 Not found** – Invalid id

Role _job_write

Abort job

DELETE /job/ (*job-id*)
Does not delete the job, but aborts it.

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 ABORT_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** – Optional string, reason for cancellation.

- **cleanup** –
  - `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.

  New in version 4.2.4.

Status Codes

- **404 Not found** – Invalid id

Role _job_write

Create duplicate job

POST /job/ (*job-id*) **/re-run**

Retrieves an existing job, duplicates it and starts the duplicated version.

Changed in version 4.2.5: The job priority now defaults to the priority of the existing job. Previously the priority was always MEDIUM.

Query Parameters

- **priority** – The priority of the new job. If no priority is specified then the priority of the existing job will be used.

  New in version 4.2.5.

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.

New in version 4.6.

Query Parameters

- **type** – The job type name. Mandatory parameter.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Status Codes

- **400** – Invalid job type name

Produces

- **application/xml, application/json** – JobDocument

Role _job_write

16.12.2 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.

Get a list of all active 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

Get a list of all job problems that affects a specific 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

16.12.3 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.

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.

ABORT_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 an EJB.)

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.

New in version 4.0.

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 *Get a list of all job problems that affects a specific 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.

16.12.4 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.

New in version 4.0.1.

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.

Get list of job types

GET /jobtype
Get list of job types

Produces

- **application/xml, application/json** – URIListDocument
- **text/plain** – list of job types

16.12.5 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.

Since 4.2.8

cerifyPriority
The priority to assign jobs created in Cerify. Either `LOW`, `MEDIUM` or `HIGH`.

Default `LOW`

Since 4.2.8

16.13 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.

16.13.1 Managing libraries

Retrieve a list of all libraries

GET /library
Retrieves a list of the ids of all known libraries.

Matrix Parameters

- **first** – Integer, from resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** – Integer, set a limit on maximum number of hits. Default `100`.
- **autoRefresh** – Optional boolean value, only list libraries with the specified auto refresh status.

  New in version 4.2.3.
- **frequencyFrom** – Optional integer value, only list libraries whose update frequency is greater than it.

  New in version 4.2.3.
- **frequencyTo** – Optional boolean value, only list libraries whose update frequency is less than it.

  New in version 4.2.3.
- **updateMode** – Optional string value, only list libraries with the specified update mode.

  New in version 4.2.3.

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** – 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** –
  - `true` - Start a DELETE_LIBRARY job that removes the library.
  - `false` (default) - Remove the library immediately.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

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
```

16.13.2 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>
   <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

New in version 4.0.3.

PUT /library/ (*library-id*) **/settings**
Update the settings of a library.

Accepts

- **application/xml, application/json** – LibrarySettingsDocument

Role _library_write

16.13.3 Library content

Retrieve library content

GET /library/ (*library-id*)
Returns the items together with any requested data.

Matrix Parameters

- **starttc** –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.

Query Parameters

- **first** – Integer, from resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** – Integer, set a limit on maximum number of hits. Default `100`.
- **count** –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **noauth-url** –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **baseURI** – Which base URI to use for the thumbnail URLs.
- **p** – Comma separated list of paths specifying the content to include. Overrides the content and filter parameters. See *Content paths*.

  New in version 4.5.
- **content** – Comma-separated list of the types of content to retrieve, possible values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** – 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`
- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.
  - *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** – 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** – 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** – 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** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **includeValues** – Return the value enumeration for each metadata field.
- **conflict** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **terse** –
  - `yes` - Return metadata in terse format (see *Retrieve terse metadata schema*)
  - `no` (default) - Return metadata in verbose format
- **defaultValue** –
  - `true` - For unset fields, return default values. See *Default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** –

  - `true` (default) - Include transient metadata, see *Transient metadata*.
  - `false` - Do not include transient metadata in response.
- **`revision`** – Optional revision, specifying which metadata to display. Only used if requesting a single item or collection.
- **`mergedType`** – The type of operation to check for.
- **`mergedPermission`** – The lowest required permission level.
- **`mergedExtradata`** – Any possible extradata.
- **`uriType`** – Optional comma-separated list of format types (container format) to return.
- **`scheme`** – URI scheme to return.
- **`storageType`** – Only return URIs for files from storages of this type. See *Storages*.
- **`methodType`** – Optional type of access method. See *Storages*.
  - `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`** – Optional metadata used with storage method. See *Storages*.
- **`tag`** – A URI parameter, see *Get item URI*.
- **`version`** – Optional integer, 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`** – A URI parameter, see *Get item URI*.

Produces

- **`application/xml, application/json`** – ItemListDocument containing the items together with the requested data.

Role _library_read

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 a specific 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 a specific 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** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Accepts

- **application/xml, application/json** – MetadataDocument the metadata to apply to the items.

Produces

- **application/xml, application/json** – JobDocument

Role _library_write

16.13.4 Listing library items in batch

New in version 4.5.

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** – The URI to output the CSV file to.
- **outputFormat** – Specifies the output format. Valid values are `xml` (default) and `csv`.
- **field** – A comma-separated list of metadata fields to include in the result.
- **data** – Specifies any additional data that should be included with the metadata fields.
- **p** – Comma-separated list of paths specifying the content to include. Overrides the field and data parameters. Only supported for XML output. See *Content paths*.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

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>
```

16.14 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.

16.14.1 Version and license

Get version and license information

GET /version
Display your license allowance and current system usage.

Produces

- **application/xml, application/json** – VersionDocument
- **text/plain** – The version details in a informational text format.

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.

Retrieve license file

GET /license
Retrieves the contents of the installed license file.

Produces

- **text/plain** – The contents of the license file.

16.14.2 Slave management and monitoring

Install slave license on master node

PUT /license/slave

Query Parameters

- **path** – The name of the slave license file.

Produces

- **application/xml, application/json** – SlaveLicenseDocument

Install slave license on master node

POST /license/slave

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.

New in version 4.0.3.

Produces

- **application/xml, application/json** – VersionDocument

Retrieve slave license file

GET /license/slave/ (*id*) **/license**
Returns the slave license for a specific slave.

Produces

- **application/xml, application/json** – SlaveLicenseDocument

Remove slave instance

DELETE /license/slave/ (*id*)
Removes the slave with the given id.

New in version 4.0.3.

List all installed slave licenses

GET /license/slave/license
Returns the `id` and `SlaveIdentifier` of all installed slave license on a master

Produces

- **application/xml, application/json** – SlaveLicenseListDocument

List installed slave licenses by id

GET /license/slave/license/ (*id*)
Returns the detail of an installed slave license with the given id

Produces

- **application/xml, application/json** – SlaveLicenseDocument

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*.

16.15 Metadata

16.15.1 Auto-projection rules

Working with automatic projection rules

Retrieve all automatic projection rules

GET /auto-projection

Produces

- **application/xml, application/json** – URIListDocument
- **text/plain** – CRLF-delimited list of names

Role _auto_projection_read

Retrieve all disabled automatic projection rules

GET /auto-projection/disable

Produces

- **application/xml, application/json** – URIListDocument
- **text/plain** – CRLF-delimited list of names

Role _auto_projection_read

Retrieve all enabled automatic projection rules

GET /auto-projection/enable

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*)
Returns 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*)

Role _auto_projection_write

Disable an automatic projection rule

PUT /auto-projection/(*name*)**/disable**

Role _auto_projection_write

Enable an automatic projection rule

PUT /auto-projection/ (*name*) **/enable**

Role _auto_projection_write

16.15.2 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

Remove 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".

Matrix Parameters

- **start** – A *time code* that defines the start of the interval.
- **end** – 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>
```

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.

Matrix Parameters

- **start** – A *time code* that defines the start of the interval.
- **end** – A *time code* that defines the end of the interval.
- **stream** – The stream index (optional).
- **channel** – The audio channel index (optional).
- **itemTrack** – The track in the item (optional).

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.

Matrix Parameters

- **start** – A *time code* that defines the start of the interval.
- **end** – A *time code* that defines the end of the interval.

Role _metadata_write

16.15.3 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.

Retrieving the global metadata

GET /metadata
Retrieves the global metadata.

Matrix Parameters

- **interval** – 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`
- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.
  - *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** – 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** – 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** – 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** –
  - `yes` (default) - Include all metadata conflicts, unresolved.

- `no` - Return conflicts resolved according to field rules.

- **include** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **defaultValue** –
  - `true` - For unset fields, return default values. See *Default values*.
  - `false` (default) - Do not return default values.

Produces

- **application/xml, application/json** – MetadataDocument

Role _metadata_global_read

Modifying the global metadata

PUT /metadata

Modifies the global metadata. This resource shares the same query and matrix parameters as the item metadata resource.

Query Parameters

- **revision** – The known revision. If not specified, the change set will attempt to override existing change sets.
- **skipForbidden** – Skip fields or groups that the user doesn't have write access to.

Accepts

- **application/xml, application/json** – MetadataDocument

Produces

- **application/xml, application/json** – MetadataDocument

Retrieving 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>
```

Removing 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
```

16.15.4 Document metadata

New in version 4.2.3.

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.

Retrieve a list of all documents

GET /document

Retrieves the list of metadata documents.

Matrix Parameters

- **first** – Optional integer. Return documents from that number in the document list. Default is `1`.
- **number** – Optional 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 by name

GET /document/ (*name*)
Retrieves the document with the specified name. This resource shares the same query and matrix parameters as the *item metadata resource*.

Matrix Parameters

- **interval** – 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`
- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.
  - *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** – 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`** – 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`** – 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`** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **`include`** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **`defaultValue`** –
  - `true` - For unset fields, return default values. See *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>
```

Create/modify a document

`PUT /document/` (*name*)

Creates a new or modifies the existing document with the specified name.

Query Parameters

- **revision** – The known revision. If not specified, the change set will attempt to override existing change sets.
- **skipForbidden** – Skip fields or groups that the user doesn't have write access to.

Accepts

- **application/xml, application/json** – MetadataDocument

Produces

- **application/xml, application/json** – MetadataDocument

Role _document_read

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>
```

View change sets

GET /document/ (*name*) **/changes**

Retrieves all change sets that have been applied to the document.

Query Parameters

- **change** – An optional parameter to retrieve a single change set.
- **first** – Optional integer. Return change sets from that number in the list of sorted change sets. Default is `1`.
- **number** – Optional 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>
```

Remove a document

DELETE /document/(*name*)

Removes the metadata document with the specified name.

Role _document_write

16.15.5 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.

Supported resources

The resources that support key-value metadata can be seen in the table below. These are referred to as `{key-value-metadata-resource}` the the definitions below.

Resource	URI	Read role	Write role
Groups	`/group/{group-name}/metadata`	`_group_read`	`_group_write`
Users	`/user/{username}/metadata`	`_admininstrator`	`_administrator`
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/{storagegroup-name}/metadata`	`_storage_group_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_group_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`

Managing key-value metadata

Retrieving 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>
```

Setting multiple key-value pairs

PUT {key-value-metadata-resource}
Sets all the specified key-value pairs.

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
```

Clearing 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
```

Retrieving the metadata for a specific key

GET {key-value-metadata-resource}/(*key*)
Retrieves the value of a specific key.

Produces

- **text/plain** – The raw string value.

Example

```
GET /user/myuser/metadata/location
```

```
London
```

Setting the value for a specific key

PUT {key-value-metadata-resource}/(*key*)
Sets the value for a specific key.

Accepts

- **text/plain** – The raw string value.

Example

```
PUT /user/myuser/metadata/location
Content-Type: text/plain

Stockholm
```

```
200 OK
```

Delete a specific key-value pair

DELETE {key-value-metadata-resource}/(*key*)
Deletes the key-value pair with the specified key.

Example

```
DELETE /user/myuser/metadata/location
```

```
200 OK
```

16.15.6 Metadata

This page describes metadata related resources for item and collection.

In the following reference, `{metadata-entity}` is one of the following:

- `/item`
- `/collection`

Changed in version 4.5: Collections now support the metadata operations listed here.

Get metadata for an entity

Get metadata

GET {metadata-entity}/(*entity-id*)**/metadata**
Returns the metadata set for an entity.

Matrix Parameters

- **projection** – (only supported in item metadata)

  Return metadata set according to projection. Default projection is `default`.

- **interval** – 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`
- **starttc** –
  - `true` - Interval is given relative to start timecode of item.
  - `false` (default) - Interval is 0-based.
- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.
  - *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** – 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** – 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`** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **`samplerate`** – 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`** – A *change-set-id*, retrieves metadata the way it looked at the given revision.
- **`terse`** – (only supported in item metadata)
  - `yes` - Return metadata in terse format (see *Retrieve terse metadata schema*)
  - `no` (default) - Return metadata in verbose format
- **`include`** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **`defaultValue`** –
  - `true` - For unset fields, return default values. See *Default values*.
  - `false` (default) - Do not return default values.
- **`from`** – (only supported in item metadata)

  A timestamp value. Return metadata starting from the specific timestamp (inclusive)
- **`to`** – (only supported in item metadata)

  A timestamp value. Return metadata until the specific timestamp (non-inclusive)

Query Parameters

- **`includeTransientMetadata`** –
  - `true` (default) - Include transient metadata, see *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

Semantics

Returns the metadata set for an entity, see *Identifiers*. This means all metadata change sets, combined, and then filtered according to matrix 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.

Examples

```
GET /item/VX-7888/metadata;field=audio-comments:comment;track=A3;interval=40-60;
↪samplerate=PAL;language=en
```

```
<MetadataListDocument>
    <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>
    <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

Add a metadata change set

PUT {metadata-entity}/(*entity-id*)**/metadata**
Sets the metadata for an entity.

Query Parameters

- **revision** – The known revision. If not specified, the change set will attempt to override existing change sets.
- **skipForbidden** – Skip fields or groups that the user doesn't have write access to.

Matrix Parameters

- **projection** – (only supported in item metadata)

  Sets metadata set according to projection. Default projection is `default`
- **output-projection** – (only supported in item metadata)

  Returns metadata according to projection. Default projection is `default`

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

Semantics

Sets the metadata for an entity (see *Identifiers*), 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.

Moving metadata

PUT {metadata-entity}/(*entity-id*)**/metadata/move**

Query Parameters

- **start** – The new start *Time codes*
- **end** – The new end *Time codes*
- **uuid** – The UUID of the element.

Status Codes

- **400** – Invalid input.
- **404** – Invalid id.

Produces

- **application/xml, application/json** – MetadataDocument

Role _metadata_write

Semantics

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.

Example

Retrieving the current metadata and checking the UUID of the top-level group element.

```
GET /item/VX-7620/metadata
```

```
<MetadataDocument>
   <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>
   <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>
```

Viewing change sets

GET {metadata-entity}/ (*entity-id*) **/metadata/changes**
Retrieves change sets that have been applied to the metadata.

Query Parameters

- **change** – An optional parameter to retrieve a single change set.
- **first** – Optional integer. Return change sets from that number in the list of sorted change sets. Default is `1`.
- **number** – Optional integer. Return at most that number of change sets. Default is all change sets.

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>user</name>
          <value user="system" timestamp="2010-03-19T09:08:09.588+01:00" change="VX-30
↪">admin</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>
```

Viewing a specific 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

Comparing change sets

New in version 4.0.

GET **{metadata-entity}/**(*entity-id*)**/metadata/changes/***changeset-id***/compareTo/previous**

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.

Query Parameters

- **valuesByUuid** – If `true` (default) then field values will be compared by uuid, if false then by the value itself.

Matrix Parameters

- **interval** – 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`
- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.
  - *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** – 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** – 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** – 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** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **include** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **defaultValue** –
  - `true` - For unset fields, return default values. See *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>
```

Modifying a single 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>
```

Modifying 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

Trimming change sets

New in version 4.2.5.

PUT {metadata-entity}/(*entity-id*)**/metadata/changes/trim**
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

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="1e941d0a-8091-4a11-b83e-09ca2bffd51d" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">
          <name>user</name>
          <value uuid="16ef4003-7ac2-4c56-bcbf-9b646aba86b2" user="admin" timestamp=
↪"2015-02-19T18:24:51.397+01:00" change="VX-816671">admin</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>
```

Deleting 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

New in version 4.1.

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*

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 several metadata entries in one request

Several entries can be modified at once using a MetadataEntryListDocument.

PUT {metadata-entity}/(*entity-id*)**/metadata/entry/**

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

Get metadata graph

GET {metadata-entity}/(*entity-id*)**/metadata/graph**
Shows fields and values and their history as a graph.

Query Parameters

- **groupBy** –
  - `change` (default) - Group fields and values by change set.

- `none` - Present fields and groups without any grouping.

Produces

- **`image/png`** –

Role _administrator

Get 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`** –
  - `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

16.15.7 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.

Note: The number of indexes returned in the `GET` command is only an estimate.

Requesting a reindex

`PUT /reindex/`(*index*)
Starts a reindex. If a reindex of the same type (item/collection/ACL/file) is already in progress, it is restarted.

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`** – One of `item`, `collection`, `acl` or `file`.

Query Parameters

- **`status`** – 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.

New in version 4.7.

- **priority** – 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>
```

Retrieving reindex information

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>
```

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>
```

16.15.8 Metadata locks

A locking container is a stateful resource that holds locks for any number of fields of the metadata of an item.

Get all containers

GET /item/ (*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

Get specific container

GET /item/ (*id*) **/metadata-lock/**
lock-id Returns information about specified locking container.

Produces

- **application/xml, application/json** – MetadataLockDocument

Role _metadata_lock_read

Create locking container

POST /item/ (*id*) **/metadata-lock**
Creates a new locking container, optionally with initial locks.

Query Parameters

- **field** – Optional comma-separated list of fields to lock.
- **timeout** – Optional integer for time-out in seconds. Default is no time-out.

Produces

- **application/xml, application/json** – MetadataLockDocument
- **text/plain** – Locking id

Role _metadata_lock_write

Add fields to locking container

PUT /item/(*id*)**/metadata-lock/**
lock-id Add new fields to the locking container and/or updates the expiry time.

Query Parameters

- **field** – Optional comma-separated list of fields to lock.
- **timeout** – Optional integer for time-out in seconds. Default is no time-out.

Produces

- **application/xml, application/json** – MetadataLockDocument
- **text/plain** – Locking id

Role _metadata_lock_write

Remove locking container and locks

DELETE /item/(*id*)**/metadata-lock/**
lock-id Remove the locking container and all locks associated with it.

Role _metadata_lock_write

16.15.9 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

Get list of fields

GET /metadata-field
Returns a list of all defined fields. See *Field identifiers*.

Query Parameters

- **includeValues** – Return the value enumeration for each field.
- **data** – An optional 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

Get field definition

GET /metadata-field/ (*field-name*)
Returns information about a specific metadata field definition.

Query Parameters

- **includeValues** – Return the value enumeration for this field.
- **data** – An optional comma-separated list of any additional data to include.

Status Codes

- **404 Not found** – The specified field could not be found.

Produces

- **application/xml, application/json** – MetadataFieldDocument

Role _metadata_field_read

Create or update field definition

PUT /metadata-field/ (*field-name*)
Creates or updates 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

Remove field definition

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

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 enumeration for 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** – Return the key with this name.
- **key-start** – Return keys starting with this prefix.
- **key-regex** – Return keys matching this regular expression.
- **value-exact** – Return keys with this value.
- **value-start** – Return keys with a value starting with this prefix.
- **value-regex** – Return keys with a value matching this regular expression.
- **hits** – 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 enumeration for 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 *Get 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 (see *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** – 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** – 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** – 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** – 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>
```

16.15.10 Metadata field access controls

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 the access control list of an item

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>
```

Add 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>
```

Remove 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
```

16.15.11 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

Get a list of known groups

GET /metadata-field/field-group
Retrieves all metadata field groups known by the system.

Query Parameters

- **content** –
  - `true` - Return the groups and their members.
  - `false` (default) - Return the group names only.
- **traverse** –
  - `true` - Traverse any sub-groups in order to retrieve the entire hierarchy.
  - `false` (default) - Only retrieves the names of the members.
- **data** – An optional comma-separated list of any additional data to include.

Produces

- **application/xml, application/json** – MetadataFieldGroupListDocument
- **text/plain** – CRLF-delimited list of field group names.

Role _metadata_field_group_read

Create a new 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

Create a new group and add fields and access to it

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.

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 an existing 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

Retrieving the fields of a group

GET /metadata-field/field-group/(*group-name*)
Retrieves the specified field group.

Query Parameters

- **includeValues** – Return the value enumeration for each field.
- **traverse** –
  - `true` - Traverse any sub-groups in order to retrieve the entire hierarchy.
  - `false` (default) - Only retrieves the names of the members.
- **data** – An optional comma-separated list of any additional data to include.

Produces

- **application/xml, application/json** – MetadataFieldGroupDocument

Role _metadata_field_group_read

Adding 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

Removing 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

Adding 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

Removing 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

Searching 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.

Matrix Parameters

- **first** – Integer, from resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** – Integer, set a limit on maximum number of hits. Default `100`.

Query Parameters

- **traverse** –
  - `true` - Traverse any sub-groups in order to retrieve the entire hierarchy.
  - `false` (default) - Only retrieves the names of the members.
- **data** – An optional comma-separated list of any additional data to include.
- **group** – Optional comma-separated list of group names to restrict search in.
- **includeValue** –
  - `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** –
  - `true` - The definition of the group is included in the result.
  - `false` (default) - The definition is not included.
- **includeSource** –
  - `true` - Information about which entity that contains the matching metadata group is included in the result.
  - `false` (default) - Entity information is not included.
- **source** –
  - `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.

  New in version 4.5.

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>
```

16.15.12 Metadata migrations

Managing migrations

List all existing migrations

GET /metadata/migration
Lists all metadata migrations defined in the system.

Role _metadata_read

Produces

- **application/xml, application/json** – MetadataSchemaMigrationListDocument

Define a new metadata migration

POST /metadata/migration
Creates a new metadata migration.

Role _metadata_write

Accepts

- **application/xml, application/json** – MetadataSchemaMigrationDocument

Produces

- **text/plain** –

View a single migration

GET /metadata/migration/ (*id*)
Shows a single metadata migration.

Role _metadata_read

Produces

- **application/xml, application/json** – MetadataSchemaMigrationDocument

16.15.13 Metadata projections

Get information about projections

Get list of projections

GET /projection
Returns a list of all defined projections.

Query Parameters

- **url** –
  - `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

Get 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

Get 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.

Create projection/set 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

Create projection/set 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

Remove 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

16.15.14 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>
```

Remove 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 particular 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
```

16.15.15 Subtitles

Vidispine is able to produce SCC and TTML files from given subtitle metadata on items.

Changed in version 4.7: Support for SCC subtitle export was added.

Exporting subtitles

Export to SCC

GET /item/ (*id*) **/metadata/export/scc**
Returns the subtitles from the item metadata in SCC format.

New in version 4.7.

Matrix Parameters

- **interval** – A list of *time spans* of subtitle metadata to export. Defaults to `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**

Matrix Parameters

- **interval** – A list of *Time span* of subtitle metadata to export. Defaults to `all`, meaning all subtitles.

Produces

- **application/xml** – The TTML XML.

Role _metadata_read

New in version 4.1.2: An optional offset can be applied to the time span by adding ":" and the offset, see example below.

Note: 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))

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/>
      </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
```

New in version 4.1.2: It is 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
```

16.16 Miscellaneous

16.16.1 Stitching images

New in version 4.0.

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** – Multiple uri instances of the images to use. For example: `uri=img1.png&uri=img2.png&uri=img3.png...`
- **geometry** – Optional 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** – The tiling strategy. Examples are `2x2` or `2x` or `x3`... If no parameter is specified, the system will try find the best alignment.
- **format** – The format of the output image, default is PNG.
- **background** – 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
```

16.16.2 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.

16.16.3 Troubleshooting

New in version 4.2.2.

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"
    }
  ]
}
```

16.16.4 WADL

A WADL file describing the API can be obtained using `GET API/application.wadl`.

GET /application.wadl

Produces

- **application/vnd.sun.wadl+xml, application/xml** – The application WADL

16.16.5 Callback

New in version 4.6.

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** – The id of the CloudConvert process
- **url** – location of the process result

- **step** – The current step of the CloudConvert process

16.17 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.

16.17.1 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, EJB, Amazon SQS (New in 4.7.) 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, EJB method, JMS queue, SQS 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>
```

Changed in version 4.6: In prior versions, all asynchronous notifications were delivered sequentially by a single thread.

HTTP

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. (New in 4.1.4.) 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>
```

EJB

Methods in EJBs 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: JNDI names must be prefixed by `vidibrain` (case insensitive).

Action parameters

The details of the EJB action are set in the action element in the notification definition.

bean The JNDI name of the bean. Mandatory parameter.

method The method name of the bean. See above for method signature. Mandatory parameter.

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>
      </ejb>
   </action>
   <trigger>
      ...
   </trigger>
</NotificationDocument>
```

JMS

JMS queues can be notified. 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 EJB action are set in the action element in the notification definition.

queueFactory The JNDI name of the JMS factory. Mandatory parameter.

queue The JNDI name of the JMS queue. Mandatory parameter.

synchronous See below. Mandatory parameter.

contentType (New in 4.2.1.) 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">
         <queueFactory>VidibrainQueueFactory</queueFactory>
         <queue>VidibrainQueue</queue>
      </jms>
   </action>
   <trigger>
      ...
   </trigger>
</NotificationDocument>
```

JavaScript

New in version 4.2.

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 EJB 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

New in version 4.7.

endpoint The endpoint of the SQS. Mandatory parameter.

queue The queue name. Mandatory parameter.

accessKey AWS accessKey. 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 secretKey; Otherwise, it should be the alias of the secret that contains the credentials to SQS.

Optional parameter.

More info about storing secrets in Vidispine.

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>
    <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>
```

16.17.2 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. (New in 4.0.)
	(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 *Get 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

New in version 4.0.

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. (New in 4.0‘‘.)
	(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 *Get 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>
```

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

New in version 4.0.

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 EJB 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

New in version 4.1.

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>
```

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>
```

16.17.3 Notifications

Most URLs that acts as resources in the RESTful 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`
- `/quota`
- `/group`

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.

Retrieve all notifications that exists within an entire resource

GET {notification-entity}/notification/
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}_notification_read

Example

```
GET /item/job/notification
Accept: application/xml
```

```
<URIListDocument xmlns="http://xml.vidispine.com/schema/vidispine">
  <uri>VX-573</uri>
</URIListDocument>
```

Create a new notification that is applied to the entire resource

POST {notification-entity}/notification/

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}_notification_write

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>
```

Deletes all notifications that exists within an entire resource

DELETE {notification-entity}/notification/
Removes all notifications that exists within the specified resource.

Status Codes

- **200 OK** – Everything went well.

Role _{notification-entity}_notification_write

Retrieve a particular notification

GET {notification-entity}/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}_notification_read

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>
```

Remove a particular notification

DELETE {notification-entity}/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}_notification_write

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.

Retrieve 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}_notification_read

Create a new 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}_notification_write

Deletes all notifications that exists within an entire resource

DELETE {notification-entity}/ (*entity-id*) **/notification/**
Removes all notifications that exists within the specified resource.

Status Codes

- **200 OK** – Everything went well.

Role _{notification-entity}_notification_write

Retrieve a particular 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}_notification_read

Remove a particular 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}_notification_write

16.18 Projects and versions

16.18.1 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>
```

16.18.2 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>
```

16.18.3 Version definitions

List the available 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

16.18.4 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

16.18.5 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

16.18.6 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** – The location of the file to inspect.
- **type** – 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>
```

16.18.7 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** – The location of the file to import.
- **type** – The file format.
- **assignId** – True if external ids should be created for the items in this project, using the ids from the project.
- **pauseFrame** – 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** – The id of the project to create a new version for.
- **uri** – The location of the file to import.
- **type** – The file format.
- **assignId** – True if external ids should be created for the items in this project, using the ids from the project.

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>
```

16.18.8 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** – The destination URI.
- **type** – The output format.
- **tag** – A comma separated list of shape tags specifying which shapes to reference in the output.
- **format** – A 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** – When set to true, any export problems will be returned and no file will be written.
- **content** – Comma-separated list of content to include in the output. Possible values are: `markers`, `subtitles`, `sequences`, `*` (default).
- **toSequence** – Export as a sequence with an item, instead of as a standalone item.

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>
```

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** – The output format.
- **tag** – A comma separated list of shape tags specifying which shapes to reference in the output.
- **format** – A 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** – When set to true, any export problems will be returned and no file will be written.
- **content** – Comma-separated list of content to include in the output. Possible values are: `markers`, `subtitles`, `sequences`, `*` (default).
- **storage** – Comma-separated list of item paths in format `storageId=path`.
- **item** – Comma-separated list of item paths in format `itemId=path`.
- **toSequence** – Export as a sequence with an item, instead of as a standalone item.

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`.

16.19 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.

16.19.1 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 existing quota rules

GET /quota/

Returns the quota rules that exist in the system.

Query Parameters

- **content** – Optional comma-separated list of addition content to retrieve. Possible values are: `external`.
- **filter** – Optional comma-separated list of key-value pairs (in the format key=value) that can be used to filter the result set. Possible key values are: `user`, `group`, `storage`, `storageGroup`, `collection`, `library` and `tag`.

  New in version 4.6.
- **exceeded** –
  - `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

16.20 Resources

16.20.1 Resource types

Retrieve list of resource types

GET /resource

Produces

- **application/xml, application/json** – ResourceTypeListDocument
- **text/plain** – CRLF-delimited list of resource type URLs (`/resource/` { **resource-type** })

Role _resource_read

16.20.2 Resources

Retrieve list of resources

GET /resource/ (*resource-type*)

Produces

- **application/xml, application/json** – ResourceListDocument
- **text/plain** – CRLF-delimited list of resource URLs (`/resource/` { **resource-type** } / { **resource-id** })

Role _resource_read

Create resources

POST /resource
POST /resource/ (*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 resource

PUT /resource/(*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

Get resource

GET /resource/(*resource-type*)**/**
resource-id

Produces

- **application/xml, application/json** – ResourceDocument

Role _resource_read

Delete resource

DELETE /resource/(*resource-type*)**/**
resource-id Deletes the resource. All connection from other resources to the resource will become invalid.

Role _resource_write

16.20.3 Resource status

Get resource status

GET /resource/(*resource-type*)**/**
*resource-id***/status**

Produces

- **text/plain** – Status string

Status Codes

- **400** – If the resource does not have a status.

Role _resource_read

16.21 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
```

16.21.1 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.

16.21.2 Managing scheduled requests

Listing all scheduled requests

GET /scheduled-request
Retrieves all known scheduled requests for the current user.

Query Parameters

- **state** – An optional parameter to 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>
```

Retrieving a specific 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>
```

Retrieving 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>user</name>
    <type>string-exact</type>
  </field>

...

</MetadataFieldListDocument>
```

Deleting all requests

DELETE /scheduled-request/
Deletes all scheduled requests for the current user.

Example

```
DELETE /scheduled-request/
```

```
200 OK
```

Deleting a specific request

DELETE /scheduled-request/(*request-id*)
Deletes the scheduled request with the specified id.

Example

```
DELETE /scheduled-request/802972
```

```
200 OK
```

16.22 Search

16.22.1 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.

Browse items and collections

GET /search
Browses items and collections.

Content Parameters See *Retrieving item information*

Matrix Parameters

- **first** – Integer, from resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** – Integer, set a limit on maximum number of hits. Default `100`.

Query Parameters

- **count** –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **p** – Comma separated list of paths specifying the content to include. Overrides the content and filter parameters. See *Content paths*.

  New in version 4.5.
- **content** – Comma-separated list of the types of content to retrieve, possible values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** – 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`
- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.

  - *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** – 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** – 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** – 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** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **includeValues** – Return the value enumeration for each metadata field.
- **conflict** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **terse** –
  - `yes` - Return metadata in terse format (see *Retrieve terse metadata schema*)
  - `no` (default) - Return metadata in verbose format
- **defaultValue** –
  - `true` - For unset fields, return default values. See *Default values*.
  - `false` (default) - Do not return default values.
- **includeTransientMetadata** –
  - `true` (default) - Include transient metadata, see *Transient metadata*.
  - `false` - Do not include transient metadata in response.
- **revision** – Optional revision, specifying which metadata to display. Only used if requesting a single item or collection.
- **mergedType** – The type of operation to check for.
- **mergedPermission** – The lowest required permission level.
- **mergedExtradata** – Any possible extradata.

- **uriType** – Optional comma-separated list of format types (container format) to return.
- **scheme** – URI scheme to return.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodType** – Optional type of access method. See *Storages*.
  - `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** – Optional metadata used with storage method. See *Storages*.
- **tag** – A URI parameter, see *Get item URI*.
- **version** – Optional integer, 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** – A URI parameter, see *Get item URI*.
- **noauth-url** –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **baseURI** – Which base URI to use for the thumbnail URLs.
- **save** –
  - `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

  New in version 4.7.

Produces

- **application/xml, application/json** – SearchResultDocument

Role _item_search

Role _collection_read

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*

Matrix Parameters

- **first** – Integer, from resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** – Integer, set a limit on maximum number of hits. Default `100`.

Query Parameters

- **count** –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **p** – Comma separated list of paths specifying the content to include. Overrides the content and filter parameters. See *Content paths*.

  New in version 4.5.
- **content** – Comma-separated list of the types of content to retrieve, possible values are `metadata`, `uri`, `shape`, `poster`, `thumbnail`, `access`, `merged-access`, `external`.
- **interval** – 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`
- **field** – 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** – Comma-separated list.
  - *group-name* - Return specified group.
  - *group-name* + - Return specified group and subgroups.

    New in version 4.1.
  - *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** – 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** – 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** – 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`** – A list of keys. Includes additional field specific data. See *Field metadata*. Additionally, if set to `type` the type definition of the field will be retrieved.
- **`includeValues`** – Return the value enumeration for each metadata field.
- **`conflict`** –
  - `yes` (default) - Include all metadata conflicts, unresolved.
  - `no` - Return conflicts resolved according to field rules.
- **`terse`** –
  - `yes` - Return metadata in terse format (see *Retrieve terse metadata schema*)
  - `no` (default) - Return metadata in verbose format
- **`defaultValue`** –
  - `true` - For unset fields, return default values. See *Default values*.
  - `false` (default) - Do not return default values.
- **`includeTransientMetadata`** –
  - `true` (default) - Include transient metadata, see *Transient metadata*.
  - `false` - Do not include transient metadata in response.
- **`revision`** – Optional revision, specifying which metadata to display. Only used if requesting a single item or collection.
- **`mergedType`** – The type of operation to check for.
- **`mergedPermission`** – The lowest required permission level.
- **`mergedExtradata`** – Any possible extradata.
- **`uriType`** – Optional comma-separated list of format types (container format) to return.
- **`scheme`** – URI scheme to return.
- **`storageType`** – Only return URIs for files from storages of this type. See *Storages*.
- **`methodType`** – Optional type of access method. See *Storages*.
  - `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`** – Optional metadata used with storage method. See *Storages*.
- **`tag`** – A URI parameter, see *Get item URI*.

- **version** – Optional integer, 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** – A URI parameter, see *Get item URI*.
- **noauth-url** –
  - `true` Return URIs that do not need authentication.
  - `false` (default) Return normal URIs
- **baseURI** – Which base URI to use for the thumbnail URLs.
- **save** –
  - `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

  New in version 4.7.

Accepts

- **application/xml, application/json** – ItemSearchDocument

Produces

- **application/xml, application/json** – SearchResultDocument

Role _item_search

Role _collection_read

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>
```

16.22.2 Search shapes

New in version 4.2.2.

Search shapes

GET /search/shape

PUT /search/shape

Searches shapes. Using GET is identical as performing a search with an empty search document.

Matrix Parameters

- **first** – Integer, from resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** – Integer, set a limit on maximum number of hits. Default `100`.

Query Parameters

- **count** –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **content** – Comma-separated list of the types of content to retrieve, possible values are `component`, `metadata`, `essenceVersion`, `tag`, `mimeType` and `*`.
- **methodType** – Return URIs from storage methods with a particular type. By default, return URLs with empty `methodType`. see *Storage methods*.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodMetadata** – Optional metadata used with storage method. See *Storages*.
- **scheme** – URI scheme to return.
- **save** –
  - `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

  New in version 4.7.

Accepts

- **application/xml, application/json** – ShapeSearchDocument

Produces

- **application/xml, application/json** – *ShapeListDocument*

Role _item_shape_read

16.22.3 Search files

New in version 4.2.2.

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/matrix parameters versus search documents. When using a search document it is also possible to perform joins.

Matrix Parameters

- **first** – Integer, from resulting list of items, start return list from specified offset. Default is `1`, start return list from beginning.
- **number** – Integer, set a limit on maximum number of hits. Default `100`.

Query Parameters

- **count** –
  - `true` (default) - Return hits in result.
  - `false` - Do not return hits in result, in order to produce results faster.
- **methodType** – Return URIs from storage methods with a particular type. By default, return URLs with empty `methodType`. see *Storage methods*.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodMetadata** – Optional metadata used with storage method. See *Storages*.
- **scheme** – URI scheme to return.
- **save** –
  - `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

  New in version 4.7.

Accepts

- **application/xml, application/json** – FileSearchDocument

Produces

- **application/xml, application/json** – FileListDocument

Role _file_read

16.22.4 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

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>
```

Auto-complete on one metadata field

New in version 4.1.

Since 4.1 auto-complete on one metadata field is supported. And 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>
```

16.22.5 Optimize index

New in version 4.1.

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 search index

POST /search/optimize

Query Parameters

- **blocking** –
  - `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

16.23 Self tests

New in version 4.0.

Changed in version 4.2: The self test has moved from `APInoauth` to `API`.

16.23.1 Running the test

Execute all tests

GET /API/selftest
Executes all the self tests.

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.

Produces

- **application/xml, application/json** – *SelfTestDocument*

Execute a specific test

GET /API/selftest/(*test-name*)
Executes the test with the specified name.

Produces

- **application/xml, application/json** – *SelfTestDocument*

16.24 Shape tags

16.24.1 Managing shape tags

Get list of known shape tags

GET /shape-tag/
Retrieves all shape tags known by the system.

Query Parameters

- **url** –
  - `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

Create a new 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 specific 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.

16.24.2 Tags of a shape

See *Tags of a shape* for how to manage the tags associated with a specific shape.

16.24.3 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

Set a 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** – The id of a job to retrieve job metadata from (optional).

Produces

- **application/xml, application/json** – TranscodePresetDocument

Role _shape_tag_read

16.25 Sites

16.25.1 Managing sites

Retrieve configured sites

GET /site
Retrieves the ids of the sites that have been configured.

Query Parameters

- **type** –
  - `all` (default) - Return the ids of all added sites.
  - `current` - Return the id of the current site.

Produces

- **application/xml, application/json** – URIListDocument

Add a new site

PUT /site/ (*site-id*)

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>
```

The only supported value for `syncPolicy` is currently `ONDEMAND`.

Retrieve a site

GET /site/ (*site-id*)
Retrieves the definition for a specific site.

Produces

- **application/xml, application/json** – SiteDefinitionDocument

Role _site_manager

16.26 Site rules

16.26.1 Managing site rules

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}`

Retrieve site rules for a specific 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

16.27 Storages

16.27.1 Auto-import rules

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

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

16.27.2 Files

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:

- `{storage-resource}/file/{file-id}`
- `{storage-resource}/file/path/{path}`
- `{storage-resource}/file/uri/{uri}`

Information about files in storage

List files in storage

New in version 4.1.2.

GET {storage-resource}/file

Query Parameters

- **id** – Optional comma-separated list of file ids to return. If not specified, all files are returned.
- **path** – Optional string
  - *path* - Return files under this sub-path to storage.
  - / (default) - Return all files.
- **ignorecase** –
  - `true` - Search file path case insensitively
  - `false` - Search file path case sensitively
- **recursive** –
  - `true` (default) - Return all files in tree.
  - `false` - Return only files directly under specified path.
- **wildcard** –
  - `true` - Allow use of wildcards in path.
  - `false` (default) - No wildcard handling of path.
- **hash** – Optional list of hash values. Only return files with specific hash value.
- **algorithm** – Optional hash algorithm. Search for hash values used by specified algorithm
- **count** –
  - `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** – Optional string to filter results by *file state*. Can be used multiple times to select several states.

  New in version 4.7.

- **methodType** – Return URIs from storage methods with a particular type. By default, return URLs with empty `methodType`. see *Storage methods*.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodMetadata** – Optional metadata used with storage method. See *Storages*.
- **scheme** – URI scheme to return.

Matrix Parameters

- **first** – Optional integer. From total list of files, start return list from specified number. Default is `0`.
- **number** – Optional integer. Return a maximum of specified number of files. Default is `10`.
- **sort** – Optional comma-separated list.
  - `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 States*
  - `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.

    New in version 4.2.1.
- **storage** – Optional list of storage ids. Return only files from specific storages. Same as `storage-id` in URL, but can be specified multiple times
- **filter** –
  - `all` (default) - Return all files
  - `item` Only return files associated with an item.
  - `noitem` Only return files not associated with any item.
- **excludeQueued** –
  - `true` (default if searching importable files) - Exclude the files that are queued for import
  - `false` (default if searching all files) - Do not exclude the files that are queued for import
- **includeItem** –
  - `true` - Return associated items, shapes, and components.
  - `false` (default) - Do not return any information about associated items, shapes, and components.

Produces

- **application/xml, application/json** – FileListDocument
- **text/plain** – CRLF-delimited list of file ids

Role _file_read

Tip: 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.

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

Get status of file in storage

GET {file-resource}

Query Parameters

- **methodType** – Return URIs from storage methods with a particular type. By default, return URLs with empty `methodType`. see *Storage methods*.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodMetadata** – Optional metadata used with storage method. See *Storages*.
- **scheme** – URI scheme to return.

Matrix Parameters

- **includeItem** –
  - `true` - Return associated items, shapes, and components.
  - `false` (default) - Do not return any information about associated items, shapes, and components.

Produces

- **application/xml, application/json** – FileDocument
- **text/plain** – *Tabbed tuples* of file id, file path, file size, file status, timestamp

Role _file_read

Upload file to storage

New in version 4.1.2.

POST {storage-resource}/file/data

Query Parameters

- **transferId** – An id to assign the transfer to be able to refer to it. Mandatory for chunked and passkey imports.
- **transferPriority** – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers (optional). A transfer with a high priority value is considered more important than a transfer with a low priority value.
- **path** – The path of the file on the storage (optional).

  New in version 4.7.
- **uri** – The absolute file URI (optional and should be relative to the URI of a storage method).

  New in version 4.7.
- **state** – The state of the file (optional).

  New in version 4.7.

Status Codes

- **400** – If the amount of data received does not match the given Content-Length header.

  New in version 4.2.3.

Accepts

- **application/octet-stream** – The raw essence data.

Produces

- **application/xml, application/json** – FileDocument

  New in version 4.7.

- **text/plain** – file id

Role _file_write

File data

Get file data

GET {file-resource}/data

Produces

- **application/octet-stream** – The raw file data.

Role _file_read

Upload file data

New in version 4.1.2.

POST {file-resource}/data

Query Parameters

- **transferId** – An id to assign the transfer to be able to refer to it. Mandatory for chunked and passkey imports.
- **transferPriority** – An integer between 1 and 1000 that indicates what priority the transfer should be given in relation to other transfers (optional). A transfer with a high priority value is considered more important than a transfer with a low priority value.

Status Codes

- **400** – If the amount of data received does not match the given Content-Length header.

  New in version 4.2.3.

Accepts

- **application/octet-stream** – The raw file data.

Produces

- **application/xml, application/json** – FileDocument

Role _file_write

Importable files

Get 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.

Query Parameters

- **auto** –
  - `true` - Return files that will be automatically imported due to auto-import rules.
  - `false` (default) - Do not return files that will be automatically imported.
- **id** – Optional comma-separated list of file ids to return. If not specified, all files are returned.
- **path** – Optional string
  - *path* - Return files under this sub-path to storage.
  - `/` (default) - Return all files.
- **ignorecase** –
  - `true` - Search file path case insensitively
  - `false` - Search file path case sensitively
- **recursive** –
  - `true` (default) - Return all files in tree.
  - `false` - Return only files directly under specified path.
- **wildcard** –
  - `true` - Allow use of wildcards in path.
  - `false` (default) - No wildcard handling of path.
- **hash** – Optional list of hash values. Only return files with specific hash value.
- **algorithm** – Optional hash algorithm. Search for hash values used by specified algorithm
- **count** –
  - `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** – Optional string to filter results by *file state*. Can be used multiple times to select several states.

  New in version 4.7.
- **methodType** – Return URIs from storage methods with a particular type. By default, return URLs with empty `methodType`. see *Storage methods*.
- **storageType** – Only return URIs for files from storages of this type. See *Storages*.
- **methodMetadata** – Optional metadata used with storage method. See *Storages*.
- **scheme** – URI scheme to return.

Matrix Parameters

- **first** – Optional integer. From total list of files, start return list from specified number. Default is `0`.
- **number** – Optional integer. Return a maximum of specified number of files. Default is `10`.
- **sort** – Optional comma-separated list.
  - `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 States*
  - `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.

    New in version 4.2.1.
- **storage** – Optional list of storage ids. Return only files from specific storages. Same as `storage-id` in URL, but can be specified multiple times
- **filter** –
  - `all` (default) - Return all files
  - `item` Only return files associated with an item.
  - `noitem` Only return files not associated with any item.
- **excludeQueued** –
  - `true` (default if searching importable files) - Exclude the files that are queued for import
  - `false` (default if searching all files) - Do not exclude the files that are queued for import

Produces

- **application/xml, application/json** – An ImportableFileListDocument describing the job.

Role _storage_read

Same query and matrix parameters as for *List files in storage*.

Importing a file from a storage

Start an import job

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** – The original filename of the file.
- **allowReimport** –
  - `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** – A comma-delimited list of external ids to assign to the item.
- **tag** – An optional list of *shape tags* to use for transcoding.
- **original** – An optional tag, if specified it 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** –
  - `true` (default) - Generate thumbnails as per defined by shape tag.
  - `false` - Disable thumbnail generation.
- **thumbnailService** – An optional identifier to which thumbnail resource that should be used.
- **createPosters** – An optional list of *time codes* to use for creating posters.
- **no-transcode** –
  - `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** –
  - `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** –
  - `true` (default) - Try to put the index tables (header) in front of the file.
  - `false` - Put header at end of file.
- **fastStartLength** – Optional estimated duration of the clip in seconds.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*
- **settings** – Pre-configured import settings. See *Import settings*

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

Start an import job of 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.

Query Parameters

- **allowReimport** –
  - 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** – A comma-delimited list of external ids to assign to the item.
- **tag** – An optional list of *shape tags* to use for transcoding.
- **original** – An optional tag, if specified it 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** –
  - true (default) - Generate thumbnails as per defined by shape tag.
  - false - Disable thumbnail generation.
- **thumbnailService** – An optional identifier to which thumbnail resource that should be used.
- **createPosters** – An optional list of *time codes* to use for creating posters.
- **no-transcode** –
  - 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** –
  - 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** –
  - true (default) - Try to put the index tables (header) in front of the file.
  - false - Put header at end of file.
- **fastStartLength** – Optional estimated duration of the clip in seconds.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is MEDIUM .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*
- **settings** – Pre-configured import settings. See *Import settings*

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

Move/copy/delete files from a storage

Create a file move/copy job

POST {file-resource}/storage/ (*target-storage-id*)
Starts a move or copy job for the specified file.

Query Parameters

- **move** –
  - `true` - Delete the original file when the copy has finished.
  - `false` - Just copy the file, and leave the original.
- **filename** – Optional filename, the desired target filename.
- **useOriginalFilename** – If set to true, the file will keep its original filename if available.
- **timeRequirement** – Option number of seconds the target file is required to exist before being moved due to storage rules etc.
- **limitRate** – (New in 4.0.) Optional integer, throttle the rate at which the transfer takes place (bytes/second).
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Produces

- **application/xml, application/json** – JobDocument

Role _file_write

Create a file delete job

DELETE {file-resource}
Starts a delete job for the specified file.

Query Parameters

- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .

- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

Produces

- **application/xml, application/json** – JobDocument

Role _file_write

Managing files

Create a file entity in the database

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** – The path of the file on the storage.
- **uri** – The absolute file URI (should be relative to the URI of a storage method).
- **state** – The state of the file. Default is `OPEN`.
- **createOnly** –
  - `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** –
  - `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.

Delete a file entity from the database

DELETE {file-resource}/entity

Deletes a file entity from the database. Does not touch the physical file.

Role _file_write

Set new file path

POST {file-resource}/path

Query Parameters

- **storage** – Optional storage id. The new storage, if omitted, the same storage.
- **path** – The new path. Required.
- **state** – Optional new state of the file. (OPEN, CLOSED, etc).
- **duplicate** –
  - `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.

  New in version 4.2.12.

Produces

- **application/xml, application/json** – FileDocument
- **text/plain** – File id

Role _file_write

Semantics

This command is used when a file is moved manually (without Vidispine), and caller wants to register the new path.

The path of file entities in Vidispine is immutable. The command above 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.

New in version 4.2.12: 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.

Set file state

PUT {file-resource}/state/ (*state*)

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>
```

Unassociate a file with an item

PUT {file-resource}/abandon
Unassociates (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** – The item from which the file in unassociated

Role _file_write

Re-index 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.

16.27.3 Storages

Managing storages

Retrieve list of storages

GET /storage

Query Parameters

- **size** – Optional 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** – Optional 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** – Optional 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** – Optional 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** – Optional 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** – Optional 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 Return all storages.
- **status** – Optional list of storage status, see *Storage states*
  - *status* - Returned storage has this status.
  - *−status* - Returned storage does not have this status.
  - default - Return all storages.
- **url** – 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 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>
  <lowWatermarkPercent>90</lowWatermarkPercent>
  <highWatermarkPercent>75</highWatermarkPercent>
  <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>
```

Get storage

GET /storage/ (*storage-id*)

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>
```

Modify 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 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** –
  - `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

Set the storage type

PUT /storage/ (*storage-id*) **/type/**

type Sets the type of the storage.

Role _storage_write

Get storage status

GET /storage/ (*storage-id*) **/status**

Produces

- **text/plain** – Status string

Role _storage_read

Get amount of free space on storage

GET /storage/ (*storage-id*) **/freespace**

Produces

- **text/plain** – Amount of free space as decimal number, between 0 and 100, inclusive

Role _storage_read

Rescanning

The `scanInterval` property can be used to control how often (in seconds) a storage is scanned. Default is 60 seconds. By calling `/rescan`, the system is forced to rescan a storage without delay.

Rescan a storage

POST /storage/ (*storage-id*) **/rescan**
Triggers a rescan of a single storage.

Storage methods

New in version 4.1: Credentials are encrypted. This means that passwords cannot be viewed through the API/server logs.

Get storage methods

GET /storage/ (*storage-id*) **/method**

Query Parameters

- **url** – Optional URL. Only return methods with this URL. Wildcards (`?`, `*`) can be used, for example `http:*`.

Matrix Parameters

- **read** –
  - `true` - Only return methods which have file read capability.
  - `false` (default) - Return methods regardless of file read capability.
- **write** –
  - `true` - Only return methods which have file write capability.
  - `false` (default) - Return methods regardless of file write capability.
- **browse** –
  - `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>
```

Add new/update 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** – Mandatory URL, method is access through this URL
- **read** –
  - `true` (default) - Method has file read capability
  - `false` - Method does not have file read capability
- **write** –
  - `true` (default) - Method has file write capability
  - `false` - Method does not have file write capability
- **browse** –
  - `true` (default) - Method has file browse capability
  - `false` - Method does not have file browse capability
- **bandwidth** – The *bandwidth* of this method in bytes per second. Default `0`.
- **type** – Optional method type, see *Storage methods*. 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
```

Get 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

Add new/update storage method

PUT /storage/ (*storage-id*) **/method/**
method-id Updates access method to the storage.

Query Parameters

- **url** – Mandatory URL, method is access through this URL
- **read** –
  - `true` - Method has file read capability.
  - `false` - Method does not have file read capability.
- **write** –
  - `true` - Method has file write capability.
  - `false` - Method does not have file write capability.
- **browse** –
  - `true` - Method has file browse capability.
  - `false` - Method does not have file browse capability.
- **bandwidth** – The *bandwidth* of this method in bytes per second. Default `0`.
- **type** – Optional method type, see *Storage methods*.

Produces

- **text/plain** – *Tabbed tuples* : id, URL, status string of method

Role _storage_write

Remove storage method

DELETE /storage/ (*storage-id*) **/method/**
method-id

DELETE /storage/ (*storage-id*) **/method;url=**
url Delete an access method to storage.

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** – 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 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`

Default (none)

`fileListBatchSize`
Maximum size of list of files that are updated in the database at the same time.

Default 100000

Since 4.0.3

`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`*.

Since 4.1

`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`*.

Since 4.2.5

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)

Since 4.1.1

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

Since 4.1.1

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

Since 4.1.2

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)

Since 4.2

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)

Since 4.2

probeFileBeforeClosing
If this metadata is set to `true`, Vidispine will attempt to read `OPEN` files before marking them as `CLOSED`. This is useful for Windows storages where the metadata is constant while a file is copied to the storage, but the file cannot be read (file is busy).

Default false

Since 4.2.1

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.

Since 4.4

Key-value method metadata

Storage methods support *key-value metadata*.

Reserved keys

sqsName
The name of Amazon SQS queue to poll for S3 file events. See *S3 Event 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`

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.

16.27.4 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.

Managing storage groups

Get a list of known 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

Remove 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

Storage group metadata

Storages groups support *key value metadata*.

16.27.5 Storage name rules

Working with storage name rules

Retrieve all 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** – 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** – The filename of the rule.

Status Codes

- **200 OK** – Rule removed successfully.

Role _storage_rule_write

16.27.6 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`

Retrieving storage rules across different resources

GET /storage-rule/
Retrieves all storage-rules that matches the given parameters.

Query Parameters

- **type** – A comma-separated list of types to retrieve, if not specified all types will be retrieved. Valid values are `ITEM`, `COLLECTION`, `LIBRARY` and `GENERIC`.
- **tag** – A 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>
   <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>
```

Retrieving 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** –
  - `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

Retrieving all storages rules

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
```

Retrieving a a specific 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>
```

Setting a specific rule

PUT {rule-resource}/(*tag-name*)

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 specific 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

16.28 Task definitions

Jobs are made up of a number of tasks that execute in a specific order.

16.28.1 Task definitions

Retrieve task definitions

GET /task-definition
Retrieves all tasks that have been defined in the system.

Query Parameters

- **type** – Optional *Job types* 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 task.

Query Parameters

- **url** –
  - `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

New in version 4.0.

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

16.28.2 Custom job types

New in version 4.6.

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** – An integer between 20000 and 30000, must be unique among job types. Mandatory parameter.

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

16.28.3 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

16.28.4 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 types*.

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

16.29 Task groups

Task groups can be used to control the resources that a specific job should use.

New in version 4.4.

16.29.1 Task groups

Retrieve task groups

GET /task-group
Retrieves all task groups that have been defined in the system.

Produces

- **application/xml, application/json** – TaskGroupListDocument

Role _administrator

Create or update task group

PUT /task-group/ (*group-name*)
Creates or updates 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 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** – The priority of the task group.
- **transcoder** – The id of the transcoder(s) to include in the group.
- **job** – 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** – 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

16.29.2 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

16.29.3 Key-value metadata

Task groups support *key-value metadata*.

16.30 Transfers

A transfer is normally started while doing a *Import using the request body*.

16.30.1 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.

16.30.2 Managing transfers

Retrieve all transfers of a specific state

GET /transfer
Returns all transfers that are in a particular state.

Query Parameters

- **state** – Optional *Transfer state* of the transfers to retrieve. Default is "TRANSFERRING".

Matrix Parameters

- **number** – Optional integer, the number of transfers to return. Default is all transfers.
- **first** – Optional 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 specific transfer

GET /transfer/ (*transfer-id*)
Retrieves a specific transfer.

Produces

- **application/xml, application/json** – A TransferDocument

Role _transfer_read

Set the priority of a transfer

PUT /transfer/ (*transfer-id*)
Sets a new priority for a specific transfer.

Query Parameters

- **transferPriority** – The desired priority.

Role _transfer_write

16.31 Users

16.31.1 Managing users

Retrieve a list of all users

GET /user
Retrieves a list of all known users.

Query Parameters

- **name** – List of user names to return information about. Default is all users.
- **first** – Start returning users from specified number. Default is `1`, the beginning of the list.
- **number** – Return at most specified number of users. Default is no limit.

Produces

- **application/xml, application/json** – UserListDocument

Role _administrator

Create a new user

POST /user
Creates a new user based on the information in the UserDocument.

Query Parameters

- **passwordType** –
  - `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 specific user

GET /user/ (*username*)
Returns a specific user.

Produces

- **application/xml, application/json** – XML/JSON, schema UserDocument

Role _administrator

Create a new 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

Disable a user

DELETE /user/ (*username*)
Disables a user with the given username, rendering that user unable to login.

Query Parameters

- **hard** – If set to `true`, the user will be removed completely. Else the user will be disabled.

  New in version 4.6.
- **notification** – See *Notifications* . (Optional)
- **notificationData** – See *Notifications* . (Optional)
- **priority** – The priority to assign to the job. Default is `MEDIUM` .
- **jobmetadata** – Additional information for the job task. See *Special job metadata values*

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

New in version 4.0.

PUT /user
Simple search of fields *username*, *realname* and metadata.

Query Parameters

- **number** – Return at most specified number of users. Default is *10*.

Accepts

- **application/xml, application/json** – UserSearchDocument

Produces

- **application/xml, application/json** – UserListDocument
- **text/plain** – CRLF-delimited list of *Tabbed tuples*: username, realname, groups*

Role _administrator

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` and `realname` are reserved to do search on `username` and `realname` fields

The boolean operators AND 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>
```

16.31.2 User information

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 _administrator

Change 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

16.31.3 User credentials

Change 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, see *Retrieve the salt of a user*.

Query Parameters

- **type** –
  - 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.

Any hashed passwords need to be salted using the salt of the user, see *Retrieve the salt of a user*.

Query Parameters

- **type** –
  - 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

Get an authentication token for a user

GET /user/ (*username*) **/token**

Creates a authentication for a user. This token can be used for calling the API without specifying username or password.

Query Parameters

- **seconds** – The duration of the token.
- **autoRefresh** –
  - `false` (default) - The token always expires after `seconds` seconds after the token was created.
  - `true` - (New in 4.2.2.) The expiration clock is reset with every API call.

Produces

- **text/plain** – The generated token.

Semantics

Calling this API resource will generate an authentication token. The username path parameter must match the calling user's credentials, unless the calling user has `_administrator` role. The token always has an expiration time.

New in version 4.2.2.

The rules for the expiration time depends on configuration property `userTokenMaxInterval` (default 60 seconds):

- **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.

Example

(Note, this call above is invoked by user `admin`)

```
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.

16.31.4 Group-to-user relations

Retrieve a list of groups a user belongs to

GET /user/ (*username*) **/groups**
Retrieves a list of all the groups a user belongs to.

Query Parameters

- **allgroups** –
  - `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** –
  - `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 _administrator

Retrieve a list of roles a user has

GET /user/ (*username*) **/roles**
Returns a list of all the roles a user has.

Produces

- **application/xml, application/json** – GroupListDocument

Role _administrator

Retrieve all the 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 _administrator

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** –
  - `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>
```

16.31.5 User/group visualization

New in version 4.1.2.

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.

Get user graph

GET /user/graph
Shows the relationships of users and groups.

Query Parameters

- **users** – Comma-separated list of users to include. Default is all users.
- **groups** – Comma-separated list of groups to include. Default is all groups.

Produces

- **image/png** –

Role _administrator

Get 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** – Comma-separated list of users to include. Default is all users.
- **groups** – Comma-separated list of groups to include. Default is all groups.

Produces

- **text/plain, text/vnd.graphviz** –

Role _administrator

16.32 Vidispine logs

16.32.1 Retrieving log files

New in version 3.3.

If your application has a custom form for reporting issues then you can collect the log files using the `vidispine-logs` resource.

Get log files

GET /vidispine-logs

Query Parameters

- **starttime** – Mandatory ISO 8601 start time of time span.
- **endtime** – Mandatory ISO 8601 end time of time span.
- **comment** – Optional detailed description of what the problem is, to be written in zip file.
- **user** – Optional comma-separated list of user names to retrieve information about.
- **storage** – Optional comma-separated list of storage ids to retrieve information about.
- **item** – Optional comma-separated list of item ids to retrieve information about.
- **job** – Optional comma-separated list of job ids to retrieve information about.
- **selftest** – If true, includes the result of a self-test. Default 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.

16.32.2 Log retrieval jobs

New in version 4.5.

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 log report job

`POST /vidispine-logs/job`

Query Parameters

- **`starttime`** – Mandatory ISO 8601 start time of time span.
- **`endtime`** – Mandatory ISO 8601 end time of time span.
- **`comment`** – Optional detailed description of what the problem is, to be written in zip file.
- **`user`** – Optional comma-separated list of user names to retrieve information about.
- **`storage`** – Optional comma-separated list of storage ids to retrieve information about.
- **`item`** – Optional comma-separated list of item ids to retrieve information about.
- **`job`** – Optional comma-separated list of job ids to retrieve information about.
- **`selftest`** – If true, includes the result of a self-test. Default false.

Produces

- **`application/xml, application/json`** – A JobDocument that describes the job.

Return all jobs

`GET /vidispine-logs/job`
Return information about all jobs that have not expired.

Produces

- **`application/xml, application/json`** – JobListDocument

Get job status

`GET /vidispine-logs/job/` (*job-id*)
Return information about specified job.

Produces

- **`application/xml, application/json`** – JobDocument

Get job result

GET /APInoauth/vidispine-logs/job/ (*job-id*) **.zip**
Returns the zip file generated by a finished job.

Delete job

DELETE /vidispine-logs/job/ (*job-id*)
Deletes a job and the report created.

16.32.3 Upload of logs to Vidispine

New in version 4.5.

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.

Get job files

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

Start 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.

Abort upload

DELETE /vidispine-logs/job/ (*job-id*) **/upload**
Aborts upload process.

16.33 Vidispine Server Agents

Vidispine server agents (VSAs) are typically handled with the vidispine-admin and vidispine-agent-admin shell tool. A few API commands are available for control, though.

New in version 4.6.

16.33.1 Managing VSAs

Retrieve all Vidispine server agents

GET /vxa
Returns all VSAs.

Produces

- **application/xml, application/json** – A VXAListDocument

Role _administrator

Retrieve a specific Vidispine 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 _administrator

Retrieve the VSA 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 _administrator

Add a VSA node

PUT /vxa/enable-ssh
Registers a new VSA node in the system.

Query Parameters

- **vsip** – The address to which the VSA should connect. Can be specified multiple times for a cluster configuration.
- **vsport** – The port to which the VSA should connect. Can be specified multiple times for a cluster configuration.
- **uuid** – Optional, the UUID of the VSA. If not set, Vidispine will assign a UUID.
- **vxaname** – Optional, the name of the VSA.

Produces

- **text/plain** – A text configuration to be added on the VSA's configuration.

Role _administrator

Delete a VSA node

DELETE /vxa/ (*uuid*)
Removes the VSA from the system

Role _administrator

16.34 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/, or here for the installed schema on your system.

16.34.1 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="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="1" maxOccurs="1"/>
                        <xs:element name="percentage" type="xs:int" minOccurs="1" 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="1" maxOccurs="1"/>
                        <xs:element name="percentage" type="xs:int" minOccurs="1" 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="1" maxOccurs="1"/>
                        <xs:element name="time" type="xs:int" minOccurs="1" 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: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: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: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="required" 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:element name="item" minOccurs="0" maxOccurs="unbounded" type="tns:
↪CollectionReorderEntryType"/>
            <xs:element name="collection" minOccurs="0" maxOccurs="unbounded" type=
↪"tns:CollectionReorderEntryType"/>
            <xs:element name="library" minOccurs="0" maxOccurs="unbounded" type="tns:
↪CollectionReorderEntryType"/>
        </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:sequence>
                    <xs:attribute name="priority" 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:sequence>
                    <xs:attribute name="superUser" use="optional" type="xs:boolean"/>
                    <xs:attribute name="priority" 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>

    <xs:complexType name="SequenceMediaType">
        <xs:sequence>
            <xs:element name="item" minOccurs="1" maxOccurs="1" type="tns:SiteIdType"/>
            <xs:element name="sourceTrack" minOccurs="1" maxOccurs="1" type="xs:int"/>
            <xs:element name="in" minOccurs="0" maxOccurs="1" type="tns:TimeCodeType"/>
            <xs:element name="out" minOccurs="0" maxOccurs="1" type="tns:TimeCodeType"/>
            <xs:element name="sourceIn" minOccurs="0" maxOccurs="1" type="tns:TimeCodeType"/>
            <xs:element name="sourceOut" minOccurs="0" maxOccurs="1" type="tns:TimeCodeType"/>
            <xs:element name="effect" type="tns:EffectType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

```
    <!-- Like TransitionType, except that it uses in and out points and has user friendly color values -->
    <xs:complexType name="SequenceTransitionType">
        <xs:sequence>
            <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: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: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: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: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>

    <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="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"/>
            <xs:element name="thumbnailPlugin" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="posterResolution" type="tns:ResolutionType" minOccurs="0
↪" maxOccurs="1"/>
            <xs:element name="posterBackground" type="xs:string" minOccurs="0"␣
↪maxOccurs="1"/>
            <xs:element name="faceDetect" type="xs:boolean" minOccurs="0" maxOccurs="1
↪"/>
            <xs:element name="preserveEDL" type="xs:boolean" minOccurs="0" maxOccurs=
↪"1"/>
            <xs:element name="addClipName" type="xs:boolean" minOccurs="0" maxOccurs=
↪"1"/>
```

```
1293            <xs:element name="overlay" minOccurs="0" maxOccurs="unbounded" type="tns:
↪OverlayType"/>
1294            <xs:element name="textOverlay" minOccurs="0" maxOccurs="unbounded" type=
↪"tns:TextOverlayType"/>
1295            <xs:element name="preferredSourceTag" minOccurs="0" maxOccurs="1" type=
↪"xs:string"/>
1296            <xs:element name="script" type="xs:string" minOccurs="0"/>
1297            <xs:element name="shapeMetadata" type="tns:KeyValuePairType" minOccurs="0
↪" maxOccurs="unbounded"/>
1298            <!-- Controls how the maximum time period that each chunk of samples is␣
↪going to be, only used for output of QuickTime files (MOV/MP4) -->
1299            <xs:element name="maxChunkDuration" type="tns:TimeCodeType" minOccurs="0"␣
↪maxOccurs="1"/>
1300            <xs:element name="demuxerSetting" type="tns:KeyValuePairType" minOccurs="0
↪" maxOccurs="unbounded"/>
1301            <xs:element name="muxerSetting" type="tns:KeyValuePairType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
1302        </xs:sequence>
1303    </xs:complexType>
1304
1305    <xs:complexType name="AudioTranscodePresetType">
1306        <xs:sequence>
1307            <xs:element name="codec" minOccurs="0" maxOccurs="1" type="xs:string"/>
1308            <xs:element name="bitrate" minOccurs="0" maxOccurs="1" type="xs:int"/>
1309            <xs:element name="framerate" minOccurs="0" maxOccurs="1" type="tns:
↪TimeBaseType"/>
1310            <xs:element name="channel" minOccurs="0" maxOccurs="unbounded" type="xs:
↪int"/>
1311            <xs:element name="stream" minOccurs="0" maxOccurs="unbounded" type="xs:int
↪"/>
1312            <xs:element name="preset" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded" />
1313            <xs:element name="noAudio" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
1314            <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
1315            <xs:element name="mix" type="tns:AudioTranscodePresetMixType" minOccurs="0
↪" maxOccurs="unbounded"/>
1316            <xs:element name="otif" type="tns:OtifPresetType" minOccurs="0" maxOccurs=
↪"1"/>
1317            <xs:element name="monoFile" type="xs:boolean" minOccurs="0" maxOccurs="1"/
↪>
1318            <xs:element name="allChannel" type="xs:boolean" minOccurs="0" maxOccurs="1
↪"/>
1319            <xs:element name="output" type="tns:AudioOutputType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
1320        </xs:sequence>
1321    </xs:complexType>
1322
1323    <xs:complexType name="AudioOutputType">
1324        <xs:sequence>
1325            <xs:element name="format" type="xs:string" minOccurs="0" maxOccurs="1"/>
1326            <xs:element name="codec" minOccurs="0" maxOccurs="1" type="xs:string"/>
1327            <xs:element name="bitrate" minOccurs="0" maxOccurs="1" type="xs:int"/>
1328            <xs:element name="framerate" minOccurs="0" maxOccurs="1" type="tns:
↪TimeBaseType"/>
1329            <xs:element name="channel" minOccurs="0" maxOccurs="unbounded" type="xs:
↪int"/>
1330            <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: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="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>
```

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="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="user" type="xs:string" 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: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: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: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: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: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: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: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>
    <!-- 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: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: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: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="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: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: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="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: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="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>
```

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: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: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="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>
```

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="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="tns:SiteIdType" 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>
```

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="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:sequence>
    </xs:complexType>
```

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="1" 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: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>
            <xs:element name="userWorkspaceUrl" type="xs:anyURI"></xs:element>
            <xs:element name="mxfServerWorkspacePath" type="xs:string"></xs:element>
            <xs:element name="mxfServerUserId" type="xs:integer"></xs:element>
            <xs:element name="mxfServerPathToStorage" type="xs:anyURI"></xs:element>
            <xs:element name="databaseName" type="xs:string"/>
            <xs:element name="storageId" type="tns:SiteIdType"></xs:element>
            <xs:element name="metadata" type="tns:SimpleMetadataType" minOccurs="0" maxOccurs="1" />
            <xs:element name="description" type="xs:string" minOccurs="0" />
            <xs:element name="atomShapes" type="xs:string" minOccurs="0" />
            <xs:element name="importShapes" type="xs:string" minOccurs="0" />
            <xs:element name="detectAtom" type="xs:boolean" minOccurs="0" />
            <xs:element name="enforceQuota" type="xs:boolean" minOccurs="0" />
            <xs:element name="fileImportPattern" type="xs:string" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SigniantType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="tag" type="xs:string" />
            <xs:element name="url" type="xs:anyURI" />
            <xs:element name="username" type="xs:string" />
            <xs:element name="password" type="xs:string" />
            <xs:element name="description" type="xs:string" minOccurs="0" />
        </xs:sequence>
```

```
    </xs:complexType>

    <xs:complexType name="NetworkType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="netmask" type="xs:anyURI"></xs:element>
            <xs:element name="bandwidth" minOccurs="0" type="xs:long"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ThumbnailServiceType">
        <xs:sequence maxOccurs="1" minOccurs="1">
            <xs:element name="path" type="xs:string"></xs:element>
            <xs:element name="mode" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="LDAPImportType">
        <xs:sequence>
            <xs:element name="interval" type="xs:long" minOccurs="0" maxOccurs="1"/>
            <xs:element name="importOrganizationalUnits" type="xs:boolean" minOccurs=
↪"0" maxOccurs="1"/>

            <xs:sequence minOccurs="0" maxOccurs="1">
                <xs:element name="plugin" type="xs:string" minOccurs="1" maxOccurs="1
↪"/>
                <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="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>
```

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:choice>
        </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>
```

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>
```

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: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"/>
            <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:sequence>
    </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="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: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="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="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>
```

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: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>
```

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: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: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="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="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" />
            <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="required"/>
                <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="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: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>
```

```
4186
4187        <xs:complexType name="ExportTemplateSequence">
4188            <xs:complexContent>
4189                <xs:extension base="tns:ExportTemplateType">
4190                    <xs:sequence>
4191                        <xs:element name="document" type="tns:ExportTemplateTextContent"␣
    ↪minOccurs="0" maxOccurs="1"/>
4192                        <xs:element name="generate" type="xs:boolean" minOccurs="0"␣
    ↪maxOccurs="1"/>
4193                    </xs:sequence>
4194                </xs:extension>
4195            </xs:complexContent>
4196        </xs:complexType>
4197
4198        <xs:simpleType name="ExportTemplateScript">
4199            <xs:restriction base="xs:string"/>
4200        </xs:simpleType>
4201
4202        <xs:complexType name="ExportTemplateTextContent" mixed="true">
4203            <xs:group ref="tns:ExportTemplateChoiceGroup" minOccurs="0" maxOccurs=
    ↪"unbounded"/>
4204            <xs:attribute name="scripttags" type="xs:boolean" use="optional"/>
4205        </xs:complexType>
4206
4207        <xs:complexType name="ExportTemplateText">
4208            <xs:complexContent>
4209                <xs:extension base="tns:ExportTemplateType">
4210                    <xs:sequence>
4211                        <xs:element name="content" minOccurs="0" maxOccurs="1" type="tns:
    ↪ExportTemplateTextContent"/>
4212                        <xs:element name="xslt" type="xs:string" minOccurs="0" maxOccurs=
    ↪"1"/>
4213                        <xs:element name="script" type="xs:string" minOccurs="0"␣
    ↪maxOccurs="1"/>
4214                        <xs:element name="xml" minOccurs="0" maxOccurs="1">
4215                            <xs:complexType>
4216                                <xs:sequence>
4217                                    <xs:any minOccurs="1" maxOccurs="1" processContents=
    ↪"lax"/>
4218                                </xs:sequence>
4219                                <xs:attribute name="indent" type="xs:int" use="optional"/>
4220                            </xs:complexType>
4221                        </xs:element>
4222                    </xs:sequence>
4223                    <xs:attribute name="charset" type="xs:string" use="optional"/>
4224                </xs:extension>
4225            </xs:complexContent>
4226        </xs:complexType>
4227
4228        <xs:complexType name="ExportTemplateExternal">
4229            <xs:complexContent>
4230                <xs:extension base="tns:ExportTemplateType">
4231                    <xs:sequence>
4232                        <xs:element name="uri" type="xs:anyURI" minOccurs="0" maxOccurs=
    ↪"unbounded"/>
4233                    </xs:sequence>
4234                </xs:extension>
4235            </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>

</xs:schema>
```

16.34.2 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>
```

```
88
89      <!-- Stuff for Shape starts here -->
90      <xs:complexType name="ResolutionType">
91          <xs:sequence>
92              <xs:element name="width" type="xs:unsignedInt"/>
93              <xs:element name="height" type="xs:unsignedInt"/>
94          </xs:sequence>
95      </xs:complexType>
96      <xs:complexType name="RationalType">
97          <xs:sequence>
98              <xs:element name="numerator" type="xs:int"/>
99              <xs:element name="denominator" type="xs:int"/>
100         </xs:sequence>
101     </xs:complexType>
102     <xs:complexType name="TimeBaseType">
103         <xs:complexContent>
104             <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
↪"tns:RationalType"/>
105         </xs:complexContent>
106     </xs:complexType>
107     <xs:complexType name="FrameRateType">
108         <xs:complexContent>
109             <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
↪"tns:RationalType"/>
110         </xs:complexContent>
111     </xs:complexType>
112     <xs:complexType name="TimeCodeType">
113         <xs:sequence>
114             <xs:element name="samples" type="xs:long"/>
115             <xs:element name="timeBase" type="tns:TimeBaseType"/>
116         </xs:sequence>
117     </xs:complexType>
118     <xs:complexType name="TimeIntervalType">
119         <xs:sequence>
120             <xs:element name="start" type="tns:TimeCodeType" minOccurs="0"/>
121             <xs:element name="end"   type="tns:TimeCodeType" minOccurs="0"/>
122         </xs:sequence>
123     </xs:complexType>
124     <xs:complexType name="AspectRatioType">
125         <xs:sequence>
126             <xs:element name="horizontal" type="xs:unsignedInt"/>
127             <xs:element name="vertical" type="xs:unsignedInt"/>
128         </xs:sequence>
129     </xs:complexType>
```

ComponentListDocument

```
131     <xs:element name="ComponentListDocument" xmlns:tns="http://xml.vidispine.com/
↪schema/vidispine" type="tns:ComponentListType" />
132
133     <xs:complexType name = "ComponentListType">
134         <xs:sequence>
135             <xs:element name="component" minOccurs="0" maxOccurs="unbounded" type=
↪"tns:ComponentType" />
136         </xs:sequence>
137     </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>


    <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: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="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"/>
            <xs:element name="timeBase" type="tns:TimeBaseType" minOccurs="0"/>

            <!-- Profile/presets to use.
                 MainConcept examples: "ipod", "baseline", "main", "high".
                 For libavcodec, see presets/ directory. Examples: "main", "normal",
↪"hq".
            -->
            <xs:element name="preset" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>

            <!-- Edit Decision List -->
            <xs:element name="edl" type="tns:EDLType" minOccurs="0"/>

            <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0"␣
↪maxOccurs="unbounded"/>

            <xs:element name="objectTracking" type="tns:ComplexJobOTIFType" minOccurs=
↪"0" maxOccurs="1"/>

            <xs:element name="metadata" type="tns:KeyValuePairType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ComplexJobAudioOutputType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
↪"tns:ComplexJobOutputType">
                <xs:sequence>
                    <!-- See generateThumbnails/uri -->
                    <xs:element name="thumbnailUri" type="xs:anyURI" minOccurs="0"␣
↪maxOccurs="unbounded"/>    <!-- If set, upload thumbnail to specified URIs -->
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="OverlayType">
        <xs:sequence>
            <!-- URI for image to overlay
                 Should use a pixel format suitable for alpha blending,
                 like 8-bit RGBA.
            -->
            <xs:element name="uri" type="xs:anyURI" maxOccurs="unbounded"/>
            <xs:element name="range" type="tns:SequenceRangeType" maxOccurs="unbounded
↪"/>
            <xs:element name="id" minOccurs="0" maxOccurs="1" type="tns:SiteIdType"/>

            <!-- Coordinates in video to place overlay at. Can be negative -->
            <xs:element name="x" type="xs:int"/>
            <xs:element name="y" type="xs:int"/>

            <!-- Optional: time interval to perform overlay in -->
            <xs:element name="interval" type="tns:TimeIntervalType" minOccurs="0"/>
```

```
            <xs:element name="opacity" minOccurs="0" maxOccurs="1" type="xs:int"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="TextOverlayType">
        <xs:sequence>
            <xs:element name="text" type="tns:TextRenditionType"/>

            <!-- Optional: time interval to perform overlay in -->
            <xs:element name="interval" type="tns:TimeIntervalType" minOccurs="0"/>
            <xs:element name="opacity" minOccurs="0" maxOccurs="1" type="xs:int"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ComplexJobVideoOutputType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
↪"tns:ComplexJobOutputType">
                <xs:sequence>
                    <xs:element name="scaling" minOccurs="0" maxOccurs="1" type="tns:
↪ScalingType"/>

                    <!-- Deprecated. Use <scaling> instead -->
                    <xs:element name="resolution" type="tns:ResolutionType" minOccurs=
↪"0"/>

                    <!-- Pixel format to use, like "yuv420p", "yuv422p", "uyvy422"␣
↪etc. -->
                    <xs:element name="pixelFormat" type="xs:string" minOccurs="0"/>

                    <xs:element name="gopSize" type="xs:unsignedShort" minOccurs="0"/>
                    <xs:element name="maxBFrames" type="xs:unsignedShort" minOccurs="0
↪"/>

                    <!-- rc_buffer_size, size of VBV buffer -->
                    <xs:element name="rcBufferSize" type="xs:unsignedInt" minOccurs="0
↪"/>

                    <!-- 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.
```

```
763
764         The transcoder will always double-check the resulting dimensions and PAR
765         against the desired DAR. If there's a mismatch, the job will fail. If
766         you want to force the transcoder to accept your settings, set targetDAR
767         manually to the resulting DAR.
768     -->
769     <xs:complexType name="ScalingType">
770         <xs:sequence>
771             <xs:element name="width" minOccurs="0" maxOccurs="1" type="xs:unsignedInt
↪"/>
772             <xs:element name="height" minOccurs="0" maxOccurs="1" type="xs:unsignedInt
↪"/>
773
774             <!-- Specifies the number of pixels to crop out of each side.
775                  Be careful when cropping odd numbers of pixels in any dimension
776                  that is subsampled. For instance, cropping an odd number of
777                  lines in YUV 4:2:0. This will cause the chroma siting to shift.
778             -->
779             <xs:element name="top" minOccurs="0" maxOccurs="1" type="xs:int"/>
780             <xs:element name="bottom" minOccurs="0" maxOccurs="1" type="xs:int"/>
781             <xs:element name="left" minOccurs="0" maxOccurs="1" type="xs:int"/>
782             <xs:element name="right" minOccurs="0" maxOccurs="1" type="xs:int"/>
783             <xs:element name="padColor" minOccurs="0" maxOccurs="1" type="xs:string"/>
↪ <!-- HTML (#rrggbb), if crop is negative -->
784
785             <!-- Specifies rotation. If no resolution/PAR is set, the PAR
786                  and resolution of the input is inverted to (hopefully)
787                  result in a pixel-to-pixel accurate flipping.
788                  Valid values are "left", "right" and "upsidedown" for 90, -90 and
789                  180 degrees rotation, respectively
790             -->
791             <xs:element name="rotate" minOccurs="0" maxOccurs="1" type="xs:string"/>
792
793             <!-- PAR -->
794             <xs:element name="pixelAspectRatio" minOccurs="0" maxOccurs="1" type="tns:
↪AspectRatioType"/>
795
796             <!-- Desired display aspect ratio -->
797             <xs:element name="targetDAR" minOccurs="0" maxOccurs="1" type="tns:
↪AspectRatioType"/>
798         </xs:sequence>
799     </xs:complexType>
800
801     <xs:complexType name="ComplexJobSubtitleOutputType">
802         <xs:complexContent>
803             <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
↪"tns:ComplexJobOutputType">
804                 <xs:sequence/>
805             </xs:extension>
806         </xs:complexContent>
807     </xs:complexType>
808
809     <xs:complexType name="BorderType">
810         <!-- Alpha + RGB color of border -->
811         <xs:attribute name="a" type="xs:unsignedByte" use="required"/>
812         <xs:attribute name="r" type="xs:unsignedByte" use="required"/>
813         <xs:attribute name="g" type="xs:unsignedByte" use="required"/>
814         <xs:attribute name="b" type="xs:unsignedByte" use="required"/>
```

```
815
816         <!-- Width of border in pixels in display space -->
817         <xs:attribute name="width" type="xs:unsignedByte" use="required"/>
818     </xs:complexType>
819
820     <xs:complexType name="TransitionType">
821         <xs:sequence>
822             <xs:element name="duration" type="tns:TimeCodeType"/>
823             <xs:choice>
824                 <!-- SMPTE wipe code (see S258m) -->
825                 <xs:element name="wipe" type="xs:int"/>
826
827                 <!-- Other transition, like "CrossDissolve". Corresponds to Fabric's␣
↪transitionSubTypeType -->
828                 <xs:element name="transition" type="xs:string"/>
829             </xs:choice>
830             <xs:element name="horizRepeat" type="xs:int" minOccurs="0"/>
831             <xs:element name="vertRepeat" type="xs:int" minOccurs="0"/>
832
833             <!-- startPercentage and endPercentage can optionally be used to override␣
↪the normal 0-100% transition range -->
834             <xs:element name="startPercentage" type="xs:int" minOccurs="0"/>
835             <xs:element name="endPercentage" type="xs:int" minOccurs="0"/>
836
837             <!-- If set and true, reverse the direction of the wipe -->
838             <xs:element name="reverse" type="xs:boolean" minOccurs="0"/>
839
840             <xs:element name="border" type="tns:BorderType" minOccurs="0"/>
841         </xs:sequence>
842     </xs:complexType>
843
844     <!-- Generic "I want this stream from that input" type. Works for both audio and␣
↪video -->
845     <xs:complexType name="ComplexJobInputType">
846         <xs:sequence>
847             <xs:element name="id" type="xs:int"/>
848             <xs:element name="stream" type="xs:unsignedShort"/>
849
850             <!-- Optional: transition effect to use when this input is followed by␣
↪another input in a timeline -->
851             <xs:element name="transition" type="tns:TransitionType" minOccurs="0"/>
852
853             <!-- Optional: Use this to set settings for decoders -->
854             <xs:element name="setting" type="tns:KeyValuePairType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
855         </xs:sequence>
856     </xs:complexType>
857
858     <!--
859           Input type for grabbing a specific channel from an input audio stream.
860           Several of these as inputs in a connection make it possible to create a new␣
↪stream from M existing channels in N input streams.
861     -->
862     <xs:complexType name="ComplexJobAudioChannelMapInputType">
863         <xs:complexContent>
864             <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base=
↪"tns:ComplexJobInputType">
865                 <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: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>

    <xs:complexType name="ComplexJobMixInputType">
       <xs:sequence>
            <xs:element name="mix" type="tns:ComplexJobMixType" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Used to request extraction of various types of metadata from input files to be reported to a resource as BulkyMetadata -->
    <xs:complexType name="ComplexJobBulkyMetadataRequestType">
        <xs:sequence>
            <xs:element name="targetUri" type="xs:anyURI"/>
            <!-- URI to write BulkyMetadata to -->
            <xs:element name="failIfTimecodeNotPresent" type="xs:boolean" minOccurs="0"/>   <!-- If true, fail the job if we didn't find a single timecode -->
        </xs:sequence>
    </xs:complexType>

    <!-- Information needed by the PartialFileDemuxer -->
    <xs:complexType name="PartialFileDemuxerInfoType">
        <xs:sequence>
            <!-- The user can either use a full desriptor or give a URI pointing to one -->
            <xs:choice>
                <xs:element name="descriptor" type="tns:PartialFileDescriptorType"/>
```

```
                <xs:element name="descriptorLocation" type="xs:anyURI"/>
            </xs:choice>

            <!-- Offset into original file that ComplexJobType/input/uri
                 consists of. In other words, how far into the file the binary
                 blob was cut.
            -->
            <xs:element name="byteOffset" type="xs:long" minOccurs="0"/>
            <xs:element name="adjustForPTSPredecessors" type="xs:boolean" minOccurs="0
↪" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ComplexJobAtomType">
        <xs:attribute name="uri" type="xs:anyURI" use="required"/>

        <!-- generated if not set -->
        <xs:attribute name="sourcePackageID" type="tns:UMIDType" use="optional"/>
    </xs:complexType>

    <xs:complexType name="AnalyzeAudioChannelType">
        <xs:sequence>
            <xs:element name="tone" type="xs:float" minOccurs="0" maxOccurs="unbounded
↪"/>
        </xs:sequence>

        <!-- Which channel in which stream to use these tones and thresholds for -->
        <xs:attribute name="stream" type="xs:unsignedShort" use="required"/>
        <xs:attribute name="channel" type="xs:unsignedShort" use="required"/>

        <!-- Silence threshold. Linear value proportional to maximum sample value.
             In other words:
              0.0  = -inf dB
              0.001=  -30 dB (default)
              1.0  =    0 dB
        -->
        <xs:attribute name="thresh" type="xs:float" use="optional"/>
    </xs:complexType>

    <xs:complexType name="AnalyzeAudioType">
        <xs:sequence>
            <xs:element name="otif" type="tns:ComplexJobOTIFType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="AnalyzeVideoType">
        <xs:sequence>
            <xs:element name="otif" type="tns:ComplexJobOTIFType" minOccurs="0"␣
↪maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Used for analyzing input video. See transcoder tickets #82 and #83 -->
    <xs:complexType name="ComplexJobAnalyzeType">
        <xs:sequence>
            <xs:element name="channel" type="tns:AnalyzeAudioChannelType" minOccurs="0
↪" maxOccurs="unbounded"/>
```

```
            <xs:element name="audio" type="tns:AnalyzeAudioType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="video" type="tns:AnalyzeVideoType" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="metadataUri" type="xs:anyURI" use="required"/>

        <!-- Thresholds are relative to the maximum pixel value, meaning values around 0-0.1 are reasonable -->
        <xs:attribute name="blackThresh" type="xs:float" use="optional"/>
        <xs:attribute name="blackPercentage" type="xs:int" use="optional"/>
        <xs:attribute name="barsThresh" type="xs:float" use="optional"/>
        <xs:attribute name="barsPercentage" type="xs:int" use="optional"/>
        <xs:attribute name="freezeThresh" type="xs:float" use="optional"/>
        <xs:attribute name="freezeTime" type="xs:float" use="optional"/>
    </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"/>

                <!-- 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>

    <!-- 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">
```

```
        <xs:sequence>
            <xs:element name="subClip" type="tns:SubClipType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="uri" type="xs:anyURI" use="optional"/>
        <xs:attribute name="stream" type="xs:unsignedShort" use="optional"/>
        <xs:attribute name="id" type="xs:anyURI" use="optional"/>
        <xs:attribute name="track" type="xs:unsignedShort" use="optional"/>
    </xs:complexType>

    <xs:complexType name="VideoClipType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:ClipType"/>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="AudioClipType">
        <xs:complexContent>
            <xs:extension xmlns:tns="http://xml.vidispine.com/schema/vidispine" base="tns:ClipType">
                <xs:attribute name="channel" type="xs:unsignedShort" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="TextRenditionType">
        <xs:sequence>
            <xs:element name="line" type="xs:string" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="align" type="xs:string" use="optional"/>        <!-- Text alignment. May be "center" (default), "left" or "right" -->
        <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) -->
        <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) -->
        <xs:attribute name="xRel" type="xs:double" use="optional"/>                <!-- optional, overrides x if set, in units of full width -->
        <xs:attribute name="yRel" type="xs:double" use="optional"/>                <!-- optional, overrides y if set, in units of full height -->
        <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 -->
        <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 -->
        <xs:attribute name="font" type="xs:string" use="optional"/>         <!-- default = "Arial". Font to use -->
        <xs:attribute name="size" type="xs:unsignedInt" use="optional"/>    <!-- default = 12. Size of font in pixels -->
        <xs:attribute name="sizeRel" type="xs:double" use="optional"/>    <!-- optional, overrides size if set, size of font in units of full height -->
        <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"/>

        <!-- 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="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="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: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="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: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: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="required"/>
                <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: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="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: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: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: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: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" />
                        <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>
```

CHAPTER

SEVENTEEN

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.

17.1 Prerequisites

The following prerequisite software is supported in this version.

Operating systems

Vidispine server (VS):

- Ubuntu 12.04 LTS (64 bit)
- Ubuntu 14.04 LTS (64 bit)
- Ubuntu 16.04 LTS (64 bit)
- Red Hat Linux 6.5
- Red Hat Linux 7.0 (New in 4.3.1.)

Vidispine server agent (VSA):

- Ubuntu 14.04 LTS (64 bit) [1]
- CentOS 6.5 (64 bit) [1]

Databases

- PostgreSQL 9.1 & 9.3
- MySQL 5.5, 5.6 & 5.7 [1]

Java

- Java 8

[1] Vidispine Enterprise edition only.

Other

- Solr 4.10.0 (if installed manually)
- Zookeeper 3.4.6 (for HA systems) [1]

Changed in version 4.6: Support for Ubuntu 16.04 and MySQL 5.7 was added. Support for Microsoft Windows Server, GlassFish and Java 7 was removed.

17.2 Upgrade notes

17.2.1 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.

17.2.2 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.

17.2.3 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 *Standalone Vidispine* instead.
- Support for Java 7 has been discontinued. Use Java 8 instead.
- Support for Microsoft Windows Server has been discontinued.

17.2.4 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 *Standalone Vidispine* instead.

17.2.5 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.

17.2.6 Upgrading from 4.2 to 4.3

- Solr: No changes to the documents. Re-indexing is not required.

17.2.7 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.

17.2.8 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/howtos/how-to-increase-file-descriptor-limit-on-linux).

17.3 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).

17.3.1 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).

17.3.2 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).

17.3.3 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.

17.4 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).

17.4.1 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).

17.4.2 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).

17.4.3 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).

17.4.4 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).

17.4.5 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 *Standalone Vidispine* 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 *Standalone Vidispine* instead.
- Support for Java 7 has been discontinued. Use Java 8 instead.
- Support for Microsoft Windows Server has been discontinued.

17.5 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).

17.5.1 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).

17.5.2 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).

17.5.3 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).

17.5.4 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).

17.5.5 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).

17.5.6 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).

17.5.7 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.

17.5.8 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).

17.5.9 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 *Standalone Vidispine* when upgrading to Vidispine 4.3 or later.

Removed in 4.5

Support for JBoss has been discontinued. Use *Standalone Vidispine* instead.

Upgrading from 4.4

- Solr: No changes to the documents. Re-indexing is not required.
- Support for JBoss has been discontinued. Use *Standalone Vidispine* instead.

17.6 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).

17.6.1 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.

17.6.2 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).

17.6.3 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).

17.6.4 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).

17.6.5 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 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

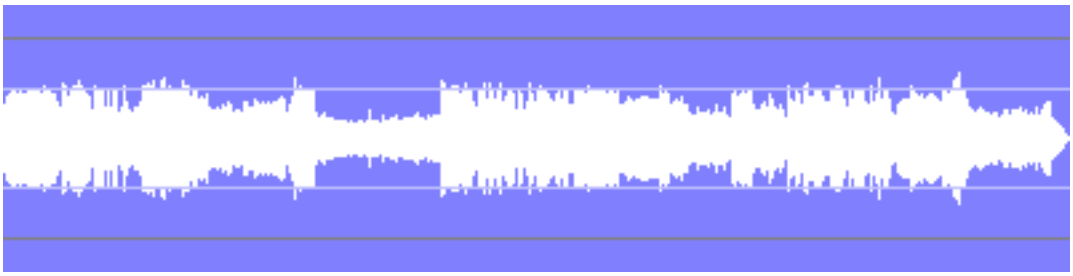
Task groups are a new entity that pairs an entity with a resource according to some rule. In this release these can be used to control the transcoders that should be used for certain types of jobs, for example to enforce that R3D jobs must use the transcoder supporting hardware decode of R3D, or to reserve 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.

17.7 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).

17.7.1 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.

17.7.2 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).

17.7.3 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).

17.7.4 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).

17.7.5 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).

17.7.6 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).

17.7.7 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).

17.7.8 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).

17.7.9 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 *Standalone Vidispine*.

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.

17.8 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).

17.8.1 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).

17.8.2 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.

17.8.3 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).

17.8.4 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).

17.8.5 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).

17.8.6 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).

17.8.7 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).

17.8.8 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).

17.8.9 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).

17.8.10 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.

17.8.11 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.

17.8.12 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.

17.8.13 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*.

17.8.14 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).

17.8.15 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).

17.8.16 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).

17.8.17 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 (http://www.vidispine.com/partner/my-software),
  - knowledge base (http://www.vidispine.com/partner/knowledge-forum-support),
  - getting started guide (http://vidispine.tenderapp.com/kb/guides/4-vidispine-getting-started-guide), and

  - discussion forum

Other tools already available on your installation

- Selftest
- Log report
- XML Schemas (XSDs)
  - xmlSchema.xsd
  - common.xsd
- Javadoc of compiled versions of the XML Schemas, useful when writing JavaScript integration code

This copy of the documentation covers Vidispine version **4.7.2** (build **4.7.2-ga80e158-17086**). You can find other versions in the Partner Portal.

/API

GET /API/selftest, 543
GET /API/selftest/(test-name), 543

/APInoauth

GET /APInoauth/callback/cloudconvert, 483
PUT /APInoauth/debug/echo, 482
POST /APInoauth/import/raw, 329
PUT /APInoauth/license/auth-info, 420
GET /APInoauth/selftest, 543
GET /APInoauth/stitch, 481
GET /APInoauth/vidispine-logs/job/(job-id).zip, 601

/application.wadl

GET /application.wadl, 483

/auto-projection

GET /auto-projection, 421
GET /auto-projection/(name), 421
PUT /auto-projection/(name), 422
DELETE /auto-projection/(name), 422
PUT /auto-projection/(name)/disable, 422
PUT /auto-projection/(name)/enable, 423
GET /auto-projection/disable, 421
GET /auto-projection/enable, 421

/collection

GET /collection, 280
PUT /collection, 299
POST /collection, 282
GET /collection/(collection-id), 282
PUT /collection/(collection-id), 293
DELETE /collection/(collection-id), 282
PUT /collection/(collection-id)/(id), 291
DELETE /collection/(collection-id)/(id), 292
GET /collection/(collection-id)/ancestor, 287
POST /collection/(collection-id)/export, 343
PUT /collection/(collection-id)/folder-name, 303
GET /collection/(collection-id)/item, 284
PUT /collection/(collection-id)/item, 288
PUT /collection/(collection-id)/map-to-folder, 303
DELETE /collection/(collection-id)/map-to-folder, 303
POST /collection/(collection-id)/order, 301
PUT /collection/(collection-id)/rename, 282
GET /collection/(id)/definition, 515
GET /collection/(id)/definition/(format), 515
PUT /collection/(id)/definition/(format), 515
DELETE /collection/(id)/definition/(format), 515
GET /collection/(id)/definition/(format)/asset, 516
PUT /collection/(id)/definition/(format)/asset, 516
DELETE /collection/(id)/definition/(format)/asset, 516
GET /collection/(id)/definition/(format)/extradata, 517
PUT /collection/(id)/definition/(format)/extradata, 517
DELETE /collection/(id)/definition/(format)/extrada 517
POST /collection/(id)/version, 514
GET /collection/(id)/version/export, 523
POST /collection/(id)/version/export, 521
GET /collection/history, 301
POST /collection/project, 513
POST /collection/project/inspect, 517

/configuration

GET /configuration, 304
GET /configuration/ftp-pool, 309
PUT /configuration/ftp-pool, 309
DELETE /configuration/ftp-pool, 309
GET /configuration/indexing, 304
PUT /configuration/indexing, 304
GET /configuration/job-pool, 306
PUT /configuration/job-pool, 307
DELETE /configuration/job-pool, 307
DELETE /configuration/job-pool/(priority), 308
GET /configuration/logreport, 310
PUT /configuration/logreport, 310
GET /configuration/metrics, 305
PUT /configuration/metrics, 305
GET /configuration/path-alias, 305
PUT /configuration/path-alias, 306
GET /configuration/properties, 310
PUT /configuration/properties, 311
GET /configuration/properties/(key), 311
PUT /configuration/properties/(key), 312
DELETE /configuration/properties/(key), 312

/conform

POST /conform, 396

/document

GET /document, 428
GET /document/(name), 429
PUT /document/(name), 430
DELETE /document/(name), 432
GET /document/(name)/changes, 431

/export-location

GET /export-location, 313
GET /export-location/(location-name), 314
PUT /export-location/(location-name), 313
DELETE /export-location/(location-name), 314
GET /export-location/(location-name)/script, 314
PUT /export-location/(location-name)/script, 314

/external-id

GET /external-id, 315
GET /external-id/(namespace-id), 315
PUT /external-id/(namespace-id), 316
DELETE /external-id/(namespace-id), 316
DELETE /external-id/id/(external-id), 318

/group

GET /group, 319
PUT /group, 320
GET /group/(group-name), 319
PUT /group/(group-name), 320
DELETE /group/(group-name), 320
GET /group/(group-name)/children, 322
GET /group/(group-name)/description, 321
PUT /group/(group-name)/description, 322
PUT /group/(group-name)/group/(child-groupname), 322
DELETE /group/(group-name)/group/(child-groupname), 323
GET /group/(group-name)/parents, 322
GET /group/(group-name)/role, 319
PUT /group/(group-name)/user/(username), 323
DELETE /group/(group-name)/user/(username), 323
GET /group/(group-name)/users, 323

/import

POST /import, 324
GET /import/access/, 272
PUT /import/access/group/(group-name), 273
DELETE /import/access/group/(group-name), 273
POST /import/placeholder, 331
POST /import/placeholder/(item-id), 335
POST /import/placeholder/(item-id)/[container,audio 332
POST /import/placeholder/(item-id)/[container,audio 333
POST /import/placeholder/(item-id)/container/adopt/ 336
POST /import/placeholder/(item-id)/raw-passkey, 336
POST /import/project, 520
POST /import/project/sequence, 518
POST /import/raw, 325
POST /import/raw-passkey, 327
GET /import/settings, 338
POST /import/settings, 338
GET /import/settings/(settings-id), 339
PUT /import/settings/(settings-id), 339
DELETE /import/settings/(settings-id), 340
POST /import/sidecar/(item-id), 337
POST /import/sidecar/(item-id)/raw, 337

/item

GET /item, 344
PUT /item, 350
GET /item/(id)/metadata-lock, 457
POST /item/(id)/metadata-lock, 457
GET /item/(id)/metadata-lock/(lock-id), 457
PUT /item/(id)/metadata-lock/(lock-id), 458
DELETE /item/(id)/metadata-lock/(lock-id), 458
GET /item/(id)/metadata/export/scc, 477
GET /item/(id)/metadata/export/ttml, 478
GET /item/(id)/relation, 362
GET /item/(id)/sequence, 364
GET /item/(id)/sequence/(format), 365
PUT /item/(id)/sequence/(format), 365
DELETE /item/(id)/sequence/(format), 365
POST /item/(id)/sequence/conform-metadata, 365
GET /item/(id)/sequence/export, 523
POST /item/(id)/sequence/export, 521
POST /item/(id)/sequence/render, 367
GET /item/(id)/shape, 368
POST /item/(id)/shape, 370
DELETE /item/(id)/shape/, 370
GET /item/(id)/shape/(shape-id), 369
DELETE /item/(id)/shape/(shape-id), 369
GET /item/(id)/shape/(shape-id)/component, 387
POST /item/(id)/shape/(shape-id)/component, 387
GET /item/(id)/shape/(shape-id)/component/(component-id), 387
POST /item/(id)/shape/(shape-id)/component/(component-id)/copy/item/(target-id)/shape/(tarq 389
POST /item/(id)/shape/(shape-id)/component/(component-id)/copy/item/(target-id)/shape/(tarq 389
POST /item/(id)/shape/(shape-id)/component/(component-id)/move/item/(target-id)/shape/(tarq 388
POST /item/(id)/shape/(shape-id)/component/(component-id)/move/item/(target-id)/shape/(tarq 388
POST /item/(id)/shape/(shape-id)/component/placeholder, 389
GET /item/(id)/shape/(shape-id)/file, 376
GET /item/(id)/shape/(shape-id)/mime/, 378
PUT /item/(id)/shape/(shape-id)/mime/(mime-type), 378
DELETE /item/(id)/shape/(shape-id)/mime/(mime-type), 378
PUT /item/(id)/shape/(shape-id)/placeholder, 375
POST /item/(id)/shape/create, 372
POST /item/(id)/shape/essence, 373
POST /item/(id)/shape/essence/raw, 374
POST /item/(id)/shape/placeholder, 375
POST /item/(id)/shape/raw, 371
GET /item/(id)/shape/version, 373
DELETE /item/(id)/shape/version/(version), 375
GET /item/(id)/shape/version/(version-number), 375
GET /item/(id)/timeline, 399
DELETE /item/(id)/timeline, 400
GET /item/(id)/timeline/(timeline-format), 400
PUT /item/(id)/timeline/(timeline-format), 400
DELETE /item/(id)/timeline/(timeline-format), 400
POST /item/(id)/timeline/(timeline-format)/conform, 397
POST /item/(id1)/relation/(id2), 363
GET /item/(item-id), 347
DELETE /item/(item-id), 349
GET /item/(item-id)/collections, 356
POST /item/(item-id)/export, 340
GET /item/(item-id)/lock, 360
PUT /item/(item-id)/lock, 361
POST /item/(item-id)/lock, 360
DELETE /item/(item-id)/lock, 361
GET /item/(item-id)/loudness, 381
PUT /item/(item-id)/loudness, 382
GET /item/(item-id)/metadata/bulky/, 423
PUT /item/(item-id)/metadata/bulky/, 423
DELETE /item/(item-id)/metadata/bulky/, 424
GET /item/(item-id)/metadata/bulky/(key), 424
PUT /item/(item-id)/metadata/bulky/(key), 424
DELETE /item/(item-id)/metadata/bulky/(key), 425
GET /item/(item-id)/posterresource, 392
PUT /item/(item-id)/posterresource, 392
PUT /item/(item-id)/re-index, 354
POST /item/(item-id)/shape/(shape-id)/analyze, 379
POST /item/(item-id)/shape/(shape-id)/export, 342
GET /item/(item-id)/shape/(shape-id)/filename, 577
PUT /item/(item-id)/shape/(shape-id)/filename/(stor 577
DELETE /item/(item-id)/shape/(shape-id)/filename/(s 578

GET /item/(item-id)/shape/(shape-id)/metadata/bulky/(job-id), 423
PUT /item/(item-id)/shape/(shape-id)/metadata/bulky/(job-id), 423
DELETE /item/(item-id)/shape/(shape-id)/metadata/bulky/(job-id), 424
GET /item/(item-id)/shape/(shape-id)/metadata/bulky/(key), 424
PUT /item/(item-id)/shape/(shape-id)/metadata/bulky/(key), 424
DELETE /item/(item-id)/shape/(shape-id)/metadata/bulky/(key), 425
GET /item/(item-id)/shape/(shape-id)/storage-rule, 579
GET /item/(item-id)/shape/(shape-id)/tag/, 377
PUT /item/(item-id)/shape/(shape-id)/tag/(tag-name), 377
DELETE /item/(item-id)/shape/(shape-id)/tag/(tag-name), 378
POST /item/(item-id)/shape/(shape-id)/thumbnail, 372
POST /item/(item-id)/shape/(shape-id)/transcode, 395
POST /item/(item-id)/shape/(shape-id)/update, 377
POST /item/(item-id)/thumbnail, 390
GET /item/(item-id)/thumbnailresource, 391
PUT /item/(item-id)/thumbnailresource, 392
POST /item/(item-id)/transcode, 394
GET /item/(item-id)/uri, 359
GET /item/(item-id)/waveform/alltracks, 386
GET /item/(item-id)/waveform/data, 383
GET /item/(item-id)/waveform/image, 383
GET /item/(item-id)/waveform/imageURI, 384
GET /item/(item-id)/waveform/values, 382
POST /item/access/, 270
DELETE /item/access/, 271
GET /item/history, 354
POST /item/list, 355

/javascript

GET /javascript/session, 401
GET /javascript/session/(id), 402
DELETE /javascript/session/(id), 402
POST /javascript/test, 401

/job

GET /job, 403
POST /job, 405
GET /job/(job-id), 404
PUT /job/(job-id), 404
DELETE /job/(job-id), 404
GET /job/(job-id)/problem, 406
POST /job/(job-id)/re-run, 405
GET /job/problem, 406

/jobtype

GET /jobtype, 408

/library

GET /library, 409
POST /library, 410
GET /library/(library-id), 412
PUT /library/(library-id), 415
DELETE /library/(library-id), 410
PUT /library/(library-id)/(item-id), 415
DELETE /library/(library-id)/(item-id), 416
POST /library/(library-id)/export, 343
PUT /library/(library-id)/item-metadata, 416
POST /library/(library-id)/list, 417
GET /library/(library-id)/settings, 411
PUT /library/(library-id)/settings, 411

/license

GET /license, 418
GET /license/slave, 419
PUT /license/slave, 419
POST /license/slave, 419
GET /license/slave/(id), 419
DELETE /license/slave/(id), 420
GET /license/slave/(id)/license, 419
GET /license/slave/license, 420
GET /license/slave/license/(id), 420

/log

GET /log, 277
GET /log/export, 279

/metadata

GET /metadata, 426
PUT /metadata, 427
GET /metadata/(uuid), 427
DELETE /metadata/(uuid), 428
GET /metadata/migration, 471
POST /metadata/migration, 471
GET /metadata/migration/(id), 472

/metadata-field

GET /metadata-field, 458
GET /metadata-field/(field-name), 459

PUT /metadata-field/(field-name), 459
DELETE /metadata-field/(field-name), 459
GET /metadata-field/(field-name)/merged-access/, 462
GET /metadata-field/(field-name)/values, 460
PUT /metadata-field/(field-name)/values, 460
GET /metadata-field/field-group, 466
PUT /metadata-field/field-group, 469
GET /metadata-field/field-group/(group-name), 468
PUT /metadata-field/field-group/(group-name), 467
DELETE /metadata-field/field-group/(group-name), 468
PUT /metadata-field/field-group/(group-name)/(field-name), 468
DELETE /metadata-field/field-group/(group-name)/(field-name), 468
GET /metadata-field/field-group/(group-name)/merged-access/, 463
PUT /metadata-field/field-group/(parent-group-name)/group/(child-group-name), 469
DELETE /metadata-field/field-group/(parent-group-name)/group/(child-group-name), 469
GET /metadata-field/field-group/merged-access/, 463
GET /metadata-field/merged-access/, 461
GET /metadata-field/terse-schema, 461

/metadata-schema

GET /metadata-schema, 474
PUT /metadata-schema, 474
DELETE /metadata-schema, 475
GET /metadata-schema/(group-name), 476
PUT /metadata-schema/(group-name), 476
DELETE /metadata-schema/(group-name), 476

/projection

GET /projection, 472
DELETE /projection/(projection-id), 473
GET /projection/(projection-id)/incoming, 473
PUT /projection/(projection-id)/incoming, 473
GET /projection/(projection-id)/outgoing, 472
PUT /projection/(projection-id)/outgoing, 473

/quota

GET /quota/, 524
POST /quota/, 524
GET /quota/(rule-id), 525
DELETE /quota/(rule-id), 526
PUT /quota/(rule-id)/evaluate, 526

/reindex

GET /reindex/(index), 456
PUT /reindex/(index), 455

/relation

GET /relation/(relation-id), 363
PUT /relation/(relation-id), 364
DELETE /relation/(relation-id), 364

/resource

GET /resource, 526
POST /resource, 526
GET /resource/(resource-type), 526
POST /resource/(resource-type), 526
GET /resource/(resource-type)/(resource-id), 527
PUT /resource/(resource-type)/(resource-id), 527
DELETE /resource/(resource-type)/(resource-id), 527
GET /resource/(resource-type)/(resource-id)/status, 527
POST /resource/(type)/(resource-id)/sync, 195

/scheduled-request

GET /scheduled-request, 528
DELETE /scheduled-request/, 530
GET /scheduled-request/(request-id), 529
DELETE /scheduled-request/(request-id), 530
GET /scheduled-request/(request-id)/response, 529

/search

GET /search, 531
PUT /search, 533
PUT /search/autocomplete, 539
GET /search/file, 538
PUT /search/file, 538
POST /search/optimize, 542
GET /search/shape, 537
PUT /search/shape, 537

/sequence

POST /sequence/export, 521
POST /sequence/render, 366

/shape-tag

GET /shape-tag/, 543
GET /shape-tag/(tag-name), 544
PUT /shape-tag/(tag-name), 544
DELETE /shape-tag/(tag-name), 544
GET /shape-tag/(tag-name)/item/(item-id)/shape/(shape-id), 545
GET /shape-tag/(tag-name)/script, 545
PUT /shape-tag/(tag-name)/script, 545
DELETE /shape-tag/(tag-name)/script, 545

/site

GET /site, 546
GET /site/(site-id), 546
PUT /site/(site-id), 546

/storage

GET /storage, 563
POST /storage, 565
GET /storage/(storage-id), 566
PUT /storage/(storage-id), 567
DELETE /storage/(storage-id), 568
GET /storage/(storage-id)/auto-import/, 550
PUT /storage/(storage-id)/auto-import/, 549
DELETE /storage/(storage-id)/auto-import/, 550
PUT /storage/(storage-id)/evacuate, 572
DELETE /storage/(storage-id)/evacuate, 572
GET /storage/(storage-id)/export, 572
GET /storage/(storage-id)/freespace, 568
GET /storage/(storage-id)/method, 569
PUT /storage/(storage-id)/method, 570
GET /storage/(storage-id)/method/(method-id), 571
PUT /storage/(storage-id)/method/(method-id), 571
DELETE /storage/(storage-id)/method/(method-id), 571
DELETE /storage/(storage-id)/method;url=(url), 571
POST /storage/(storage-id)/rescan, 569
GET /storage/(storage-id)/status, 568
PUT /storage/(storage-id)/type/(type), 568
GET /storage/auto-import/, 550
POST /storage/import, 572
GET /storage/storage-group/, 575
GET /storage/storage-group/(group-name), 576
PUT /storage/storage-group/(group-name), 576
DELETE /storage/storage-group/(group-name), 576
PUT /storage/storage-group/(group-name)/(storage-id 576
DELETE /storage/storage-group/(group-name)/(storage 577

/storage-rule

GET /storage-rule/, 578

/task-definition

GET /task-definition, 582
POST /task-definition, 583
GET /task-definition/(task-id), 583
PUT /task-definition/(task-id), 584
DELETE /task-definition/(task-id), 583
GET /task-definition/(task-id)/script, 585
PUT /task-definition/(task-id)/script, 585
GET /task-definition/(task-id)/validate, 583
GET /task-definition/jobtype/(type), 582
POST /task-definition/jobtype/(type), 584
DELETE /task-definition/jobtype/(type), 584
GET /task-definition/jobtype/(type)/graph, 585
GET /task-definition/jobtype/(type)/graph/dot, 585
GET /task-definition/jobtype/(type)/step/(step), 583
PUT /task-definition/jobtype/(type)/step/(step), 584
DELETE /task-definition/jobtype/(type)/step/(step), 583
GET /task-definition/jobtype/(type)/step/(step)/scr 585
PUT /task-definition/jobtype/(type)/step/(step)/scr 585
GET /task-definition/jobtype/(type)/step/(step)/val 583

/task-group

GET /task-group, 586
DELETE /task-group, 588
GET /task-group/(group-name), 587
PUT /task-group/(group-name), 587
DELETE /task-group/(group-name), 588
PUT /task-group/(group-name)/transcoder/(transcoder 588
DELETE /task-group/(group-name)/transcoder/(transco 588

/transfer

GET /transfer, 589
GET /transfer/(transfer-id), 589
PUT /transfer/(transfer-id), 590

/user

GET /user, 590
PUT /user, 592
POST /user, 590
GET /user/(username), 591
PUT /user/(username), 591
DELETE /user/(username), 591
GET /user/(username)/allgroups, 596
PUT /user/(username)/enable, 592
GET /user/(username)/groups, 596
PUT /user/(username)/groups, 597
PUT /user/(username)/password, 593
GET /user/(username)/password/salt, 594
POST /user/(username)/password/salt, 595
GET /user/(username)/realname, 593
PUT /user/(username)/realname, 593
GET /user/(username)/roles, 596
GET /user/(username)/token, 595
PUT /user/(username)/validate, 594
GET /user/graph, 598
GET /user/graph/dot, 599

/version

GET /version, 418

/vidispine-logs

GET /vidispine-logs, 599
GET /vidispine-logs/job, 600
POST /vidispine-logs/job, 600
GET /vidispine-logs/job/(job-id), 600
DELETE /vidispine-logs/job/(job-id), 601
POST /vidispine-logs/job/(job-id)/upload, 601
DELETE /vidispine-logs/job/(job-id)/upload, 601
GET /vidispine-logs/job/(job-id)/uri, 601

/vxa

GET /vxa, 602
GET /vxa/(uuid), 602
DELETE /vxa/(uuid), 603
GET /vxa/(uuid)/configuration, 602
PUT /vxa/enable-ssh, 602

/{access-entity}

GET {access-entity}/(item-id)/access/, 269
POST {access-entity}/(item-id)/access/, 269
GET {access-entity}/(item-id)/access/(access-id), 270
DELETE {access-entity}/(item-id)/access/(access-id), 270
POST {access-entity}/(item-id)/access/bulk, 271
DELETE {access-entity}/(item-id)/access/bulk, 272
GET {access-entity}/(item-id)/access/graph, 276
GET {access-entity}/(item-id)/access/graph/dot, 277
GET {access-entity}/(item-id)/merged-access/, 273
GET {access-entity}/(item-id)/merged-access/group, 275

/{external-id-resource}

GET {external-id-resource}, 317
DELETE {external-id-resource}, 318
PUT {external-id-resource}/(external-id), 317
DELETE {external-id-resource}/(external-id), 318

/{field-access-resource}

GET {field-access-resource}, 464
POST {field-access-resource}, 465
DELETE {field-access-resource}/(access-id), 465

/{file-resource}

GET {file-resource}, 554
DELETE {file-resource}, 560
PUT {file-resource}/abandon, 563
GET {file-resource}/data, 555
POST {file-resource}/data, 555
DELETE {file-resource}/entity, 562
POST {file-resource}/import, 557
POST {file-resource}/import/assetmap, 559
POST {file-resource}/path, 562
PUT {file-resource}/re-index, 563
PUT {file-resource}/restore, 232
PUT {file-resource}/state/(state), 562
POST {file-resource}/storage/(target-storage-id), 560

/{item-content-resource}

GET {item-content-resource}, 357

/{key-value-metadata-resource}

GET {key-value-metadata-resource}, 433
PUT {key-value-metadata-resource}, 434
DELETE {key-value-metadata-resource}, 434
GET {key-value-metadata-resource}/(key), 434
PUT {key-value-metadata-resource}/(key), 435
DELETE {key-value-metadata-resource}/(key), 435

/{metadata-entity}

GET {metadata-entity}/(entity-id)/metadata, 436
PUT {metadata-entity}/(entity-id)/metadata, 439
GET {metadata-entity}/(entity-id)/metadata/changes, 441
PUT {metadata-entity}/(entity-id)/metadata/changes/, 448
GET {metadata-entity}/(entity-id)/metadata/changes/(changeset-id), 443
PUT {metadata-entity}/(entity-id)/metadata/changes/(changeset-id), 446
DELETE {metadata-entity}/(entity-id)/metadata/changes/(changeset-id), 451
GET {metadata-entity}/(entity-id)/metadata/changes/(changeset-id)/compareTo/(from-changeset 443
GET {metadata-entity}/(entity-id)/metadata/changes/(changeset-id)/compareTo/previous, 443
PUT {metadata-entity}/(entity-id)/metadata/changes/(changeset-id)/trim, 449
PUT {metadata-entity}/(entity-id)/metadata/changes/trim, 449
PUT {metadata-entity}/(entity-id)/metadata/entity/, 454
PUT {metadata-entity}/(entity-id)/metadata/entity/(entity-uuid), 452
GET {metadata-entity}/(entity-id)/metadata/graph, 454
GET {metadata-entity}/(entity-id)/metadata/graph/dot, 455
PUT {metadata-entity}/(entity-id)/metadata/move, 439

/{notification-entity}

GET {notification-entity}/(entity-id)/notification/, 512
POST {notification-entity}/(entity-id)/notification/, 512
DELETE {notification-entity}/(entity-id)/notification/, 512
GET {notification-entity}/(entity-id)/notification/ 513
DELETE {notification-entity}/(entity-id)/notificati 513
GET {notification-entity}/notification/, 509
POST {notification-entity}/notification/, 510
DELETE {notification-entity}/notification/, 511
GET {notification-entity}/notification/(notificatio 511
DELETE {notification-entity}/notification/(notifica 512

/{rule-resource}

GET {rule-resource}, 579
PUT {rule-resource}, 580
DELETE {rule-resource}, 580
GET {rule-resource}/(tag-name), 581
PUT {rule-resource}/(tag-name), 581
DELETE {rule-resource}/(tag-name), 582

/{site-rule-entity}

GET {site-rule-entity}/site-rule, 547
POST {site-rule-entity}/site-rule/, 548
GET {site-rule-entity}/site-rule/(id), 548
PUT {site-rule-entity}/site-rule/(id), 549
DELETE {site-rule-entity}/site-rule/(id), 549

/{storage-resource}

GET {storage-resource}/file, 551
POST {storage-resource}/file, 561
POST {storage-resource}/file/data, 554
GET {storage-resource}/importable, 556

/{thumbnail-resource}

GET {thumbnail-resource}, 392
DELETE {thumbnail-resource}, 393
GET {thumbnail-resource}/(time), 393
PUT {thumbnail-resource}/(time), 393
DELETE {thumbnail-resource}/(time), 394
POST {thumbnail-resource}/(time)/export, 394

A

additionalHash
    reserved key, 573
alwaysGenerateThumbnails
    configuration property, 213
api.dataType() (api method), 225
api.delete() (api method), 225
api.get() (api method), 225
api.getInfo() (api method), 226
api.input() (api method), 225
api.path() (api method), 225
api.post() (api method), 225
api.put() (api method), 225
api.queryParam() (api method), 225
api.rich() (api method), 226
api.timeout() (api method), 225
api.user() (api method), 225
apiNoauthUri
    configuration property, 209
apiUri
    configuration property, 209
auditTrailPurgingDirectory
    configuration property, 218
auditTrailPurgingTime
    configuration property, 217
azureSasValidTime
    configuration property, 214

B

bulkyMetadataKeysToIgnore
    configuration property, 218

C

cerifyPriority
    job metadata key, 408
changeLogForcePurgingTime
    configuration property, 218
changeLogPurgingTime
    configuration property, 218
checksumMode
    job metadata key, 408
closeLimit
    reserved key, 573
clusterName
    configuration property, 209
com.vidispine.credentials.dir, 130
com.vidispine.site, 7
compressDocumentMessages
    configuration property, 217
concurrentJobs
    configuration property, 212
configuration property
    alwaysGenerateThumbnails, 213
    apiNoauthUri, 209
    apiUri, 209
    auditTrailPurgingDirectory, 218
    auditTrailPurgingTime, 217
    azureSasValidTime, 214
    bulkyMetadataKeysToIgnore, 218
    changeLogForcePurgingTime, 218
    changeLogPurgingTime, 218
    clusterName, 209
    compressDocumentMessages, 217
    concurrentJobs, 212
    defaultIngestStorage, 212
    disableATime, 216
    disableMetadataSchema, 211
    disableSequenceChecker, 218
    disableSiteCrunching, 209
    disableThumbnailGeneration, 213
    enableTranscoderHashing, 214
    enableTranscoderTransfer, 216
    fileGrowingTimeout, 216
    fileHashAlgorithm, 214
    fileHashExecutionTime, 217
    fileHierarchy, 215
    fileNotGrowingTimeout, 217
    fileTempKeyDuration, 214
    firstLastModifiedAsCreationTime, 215
    indexCollectionItemOrder, 211
    indexFieldGroups, 211
    indexTimespans, 211
    itemDeleteExecutionTime, 217
    itemDeleteInterval, 217

itemDeleteIntervalShort, 217
jobExclusiveStepMaxWait, 212
jobPurgingDirectory, 218
jobPurgingTime, 218
jobRetryCount, 212
keepEmptyDirectories, 213
keepMissingFiles, 213
ldapAuthentication, 212
legacyTransientFieldTypes, 211
libraryExpireTime, 216
libraryUpdateInterval, 216
localFSTimeData, 215
maxSearchResults, 211
maxTranscoderUnavailableTime, 218
mediaCheckInterval, 213
parseFileMetadata, 212
parseXMP, 213
passwordHashAlgorithm, 212
s3ConcurrentParts, 214
s3ConnectionTimeout, 214
s3MaxErrorRetry, 214
s3PartSize, 214
s3ProxyValidTime, 214
s3SocketTimeout, 214
signiantManagerHost, 216
signiantManagerPassword, 216
signiantManagerUser, 216
simpleImageProcessor, 213
slaveLicenseProxy, 209
solrAutoSoftCommit, 210
solrCollection, 210
solrCommitInterval, 210
solrPath, 210
solrPingAttempts, 210
solrPingTimeout, 210
solrQueryTimeout, 210
solrSoftCommitInterval, 210
solrUpdateQueueSize, 211
statsPerSecond, 215
stornextFileMetadata, 215
syncVxaDeletes, 219
syncVxaFileChanges, 219
thumbnailHierarchy, 215
transcoderNonblockingStatusInterval, 218
trustArchivedFiles, 215
useAzureProxy, 214
useLucene, 216
userTokenDefaultInterval, 212
userTokenMaxInterval, 212
userTokenRefreshInterval, 212
useS3Proxy, 214
useSegmentFiles, 215
validatexml, 209
xmpIgnoreElements, 213
    zkHost, 210
constants
    storage states, 121
    storage types, 121
context.getChannel() (context method), 139
context.getComponent() (context method), 139
context.getExtension() (context method), 139
context.getFileId() (context method), 139
context.getItem() (context method), 138
context.getJobMetadata() (context method), 138
context.getOriginalFilename() (context method), 139
context.getShape() (context method), 138
context.getStorage() (context method), 139
context.getTags() (context method), 139

D

defaultIngestStorage
    configuration property, 212
deleteFileIfNotFound
    reserved key, 574
detectRenamedFiles
    reserved key, 574
disableATime
    configuration property, 216
disableMetadataSchema
    configuration property, 211
disableSequenceChecker
    configuration property, 218
disableSiteCrunching
    configuration property, 209
disableThumbnailGeneration
    configuration property, 213

E

enableTranscoderHashing
    configuration property, 214
enableTranscoderTransfer
    configuration property, 216
environment variable
    com.vidispine.credentials.dir, 130, 219
    com.vidispine.license.dir, 219
    com.vidispine.license.tmpdir, 219
    com.vidispine.log.dir, 219
    com.vidispine.site, 7, 219
    vidispine.identifier.format, 7, 220
excludeFilter
    reserved key, 574

F

file.getAllMetadata() (file method), 230
file.getMetadata() (file method), 230
file.setMetadata() (file method), 230
fileGrowingTimeout
    configuration property, 216

fileHashAlgorithm
configuration property, 214
fileHashExecutionTime
configuration property, 217
fileHierarchy
configuration property, 215
fileListBatchSize
reserved key, 573
fileNotGrowingTimeout
configuration property, 217
fileTempKeyDuration
configuration property, 214
firstLastModifiedAsCreationTime
configuration property, 215

H

hashMode
reserved key, 573
helper.createMetadata() (helper method), 61
helper.createMetadataGroup() (helper method), 61
helper.createMetadataTimespan() (helper method), 61
helper.generateMetadataField() (helper method), 61
helper.log() (helper method), 61
helper.metadataToStr() (helper method), 61
http.uri(uri):() (http method), 226

I

indexCollectionItemOrder
configuration property, 211
indexFieldGroups
configuration property, 211
indexTimespans
configuration property, 211
itemDeleteExecutionTime
configuration property, 217
itemDeleteInterval
configuration property, 217
itemDeleteIntervalShort
configuration property, 217

J

job metadata key
cerifyPriority, 408
checksumMode, 408
lastSmpteTimeCode, 408
smpteTimeCode, 408
job.fail() (job method), 149
job.fatalFail() (job method), 149
job.getData() (job method), 149
job.getId() (job method), 149
job.log() (job method), 149
job.setData() (job method), 149
job.wait() (job method), 150
jobExclusiveStepMaxWait
configuration property, 212
jobPurgingDirectory
configuration property, 218
jobPurgingTime
configuration property, 218
jobRetryCount
configuration property, 212

K

keepEmptyDirectories
configuration property, 213
reserved key, 573
keepMissingFiles
configuration property, 213
reserved key, 573

L

lastSmpteTimeCode
job metadata key, 408
ldapAuthentication
configuration property, 212
legacyTransientFieldTypes
configuration property, 211
libraryExpireTime
configuration property, 216
libraryUpdateInterval
configuration property, 216
localFSTimeData
configuration property, 215
logger.json() (logger method), 228
logger.log() (logger method), 227
lostLimit
reserved key, 573

M

maxSearchResults
configuration property, 211
maxTranscoderUnavailableTime
configuration property, 218
mediaCheckInterval
configuration property, 213

P

parseFileMetadata
configuration property, 212
parseXMP
configuration property, 213
passwordHashAlgorithm
configuration property, 212
probeFileBeforeClosing
reserved key, 574

R

refreshInterval

reserved key, 573
refreshOnStart
reserved key, 574
reserved key
additionalHash, 573
closeLimit, 573
deleteFileIfNotFound, 574
detectRenamedFiles, 574
excludeFilter, 574
fileListBatchSize, 573
hashMode, 573
keepEmptyDirectories, 573
keepMissingFiles, 573
lostLimit, 573
probeFileBeforeClosing, 574
refreshInterval, 573
refreshOnStart, 574
sqsEndpoint, 575
sqsName, 575
statsPerSecond, 573
toAppearLimit, 573
vxaId, 574

S

s3ConcurrentParts
configuration property, 214
s3ConnectionTimeout
configuration property, 214
s3MaxErrorRetry
configuration property, 214
s3PartSize
configuration property, 214
s3ProxyValidTime
configuration property, 214
s3SocketTimeout
configuration property, 214
shell.exec() (shell method), 227
signiantManagerHost
configuration property, 216
signiantManagerPassword
configuration property, 216
signiantManagerUser
configuration property, 216
simpleImageProcessor
configuration property, 213
slaveLicenseProxy
configuration property, 209
smpteTimeCode
job metadata key, 408
solrAutoSoftCommit
configuration property, 210
solrCollection
configuration property, 210
solrCommitInterval
configuration property, 210
solrPath
configuration property, 210
solrPingAttempts
configuration property, 210
solrPingTimeout
configuration property, 210
solrQueryTimeout
configuration property, 210
solrSoftCommitInterval
configuration property, 210
solrUpdateQueueSize
configuration property, 211
sqsEndpoint
reserved key, 575
sqsName
reserved key, 575
statsPerSecond
configuration property, 215
reserved key, 573
storage states
constants, 121
storage types
constants, 121
stornextFileMetadata
configuration property, 215
syncVxaDeletes
configuration property, 219
syncVxaFileChanges
configuration property, 219

T

thumbnailHierarchy
configuration property, 215
toAppearLimit
reserved key, 573
transcoderNonblockingStatusInterval
configuration property, 218
trustArchivedFiles
configuration property, 215

U

useAzureProxy
configuration property, 214
useLucene
configuration property, 216
userTokenDefaultInterval
configuration property, 212
userTokenMaxInterval
configuration property, 212
userTokenRefreshInterval
configuration property, 212
useS3Proxy
configuration property, 214

useSegmentFiles
configuration property, 215

V

validatexml
configuration property, 209
vidispine.identifier.format, 7
vxaId
reserved key, 574

W

wrapper.getBulkyMetadata() (wrapper method), 62
wrapper.getMetadata() (wrapper method), 62
wrapper.getOldBulkyMetadata() (wrapper method), 62
wrapper.getOldMetadata() (wrapper method), 62
wrapper.getShape() (wrapper method), 62
wrapper.getShapeMetadata() (wrapper method), 62
wrapper.setMetadata() (wrapper method), 62

X

xmpIgnoreElements
configuration property, 213

Z

zkHost
configuration property, 210