VOD Interface Part 3d

Content propagation interface

History

Version	Status	Date	Author	Description
1.0	Released	28-09-2009	Jan Verhoeven	First released version
1.1	Proposal	22-04-2010	Bert van Willigen	§4.2.11: Example updated to make it XSD compliant.
1.1	Released	12-11-2010	Bert van Willigen	Status update.
1.2	Released	09-05-2011	Bert van Willigen	• Rebranding to SeaChange. • §3.4.1: Table: StartTime & Time changes into IngestStartTime & IngestStartTime respectively.
1.3	Released	20-9-2012	Ralph Janssen	• Added UNC to the protocol option for SourceUri field.
1.4	Released	12-10-2012	Ralph Janssen	• Added ProfileName
1.5	Proposed	16-01-2013	Michiel Didde	Field clarifications
1.6	Proposed	22-02-2013	Kevin Farrington	nPVR enhancements

Document Statuses

This document can have one of the following statuses:

DRAFT	Document is draft.
PROPOSAL	Document is ready for review.
RELEASED	Document has been reviewed and released by SeaChange development.

Document Version Numbering

This document has unique version numbers to unique states of the document. The version number is expressed as a *major.minor* number pair. These numbers which start at zero, are assigned in increasing order and correspond to new developments of the document. Whereby the minor number is used to convey maintenance type of changes and the major one conveys new document releases that has been gone thru a review-release cycle.

Document Change Procedure

Changes to this specification will result in a new version of this document. When this document has either the DRAFT or, PROPOSAL status, change requests must be sent to the author. In case the document has RELEASED status, change requests must be submitted to the *Change Control Board* (CCB) of SeaChange development.

SeaChange®

Contents

1 Introduction

1.1 Purpose of the document

Video On Demand (VOD) systems contain many internal- and external systems and subsystems that have to be linked to each other. The purpose of this document is to describe a generic content propagation interface for use by video server vendors.

The content propagation interface specification of the SeaChange on-demand solution will be the only specification which will be used for the implementation and deployment of this interface – any other documentation describing these interfaces will be considered to be for informational purposes only.

1.2 Scope

This document deals only with the interfaces related to the VOD infrastructure. It does not describe any head-end components that are involved in the delivery of the 'unaddressed broadcasting services' to the customer.

1.3 Changes

The interfaces in this document are specified as accurately as possible at the time of the writing. Any changes to the interfaces must be mutually agreed between SeaChange and the customer. SeaChange keeps the right to improve the interfaces. Suppliers using the interfaces will be notified of such changes.

1.4 Acronyms, Abbreviations and Terms

The next list provides an overview of acronyms and abbreviations used in this document and where they stand for.

Acronym		**Stands for**
ADI		Asset Distribution Interface
CPE		Customer Premises Equipment
CMS		Content Management System
CRM		Customer Relationship Management
DVB		Digital Video Broadcasting
EPG		Electronic Program Guide
FTP		File Transfer Protocol
FVOD		Free Video On Demand
HD		High Definition
HTTP		HyperText Transfer Protocol
ID		Identity
IMEI		International Mobile Equipment Identity
ISO		International Standards Organisation
JP(E)G		Joint Photographic (Expert) Group
MAC		Media Access Control
MD5		Message Digest Algorithm 5
MPEG		Moving Picture Experts Group
MSO		Multi System Operator
NA		Not Applicable
NPVR		Network Personal Video Recorder
REST		Representational State Transfer
RTSP		Real Time Streaming Protocol

SOAP		Simple Object Access Protocol
SPTS		Single Program Transport Stream
SRM		Session Resource Manager
STB		Set Top Box
SVOD		Subscription Video On Demand
TVOD		Transaction Video On Demand
TSTV		Time Shift TeleVision
URI		Uniform Resource Identifier
UTC		Coordinated Universal Time
VAT		Value Added Tax
VOD		Video on demand
XML		eXtensible Markup Language
XSD		XML Schema Definition
XSLT		eXtensible Stylesheet Language Transformation

Below terms are listed with their definitions, as used in this document.

Term	**Definition**
Asset	Content together with metadata.
Asset group	A bundle of related assets, e.g. the feature film, its trailer and the 'making of…'. The asset group has nothing to do with a commercial offering (products).
Banner	A headline containing image- and/or text essences
Content	A bundle of related essences, e.g., an SPTS with an audio and a video stream.
Coupon	A ticket which can be used for obtaining a free product by means of a VOD transaction
Entitlement	The right to access/view content
Essence	The data that represents pictures, sound and text; types of essence include video, audio and data of various kinds, including captions, graphics still images, text enhancements and other data as needed by each application
File drop	The process of putting a (content-)file in a dedicated part (folder) of a server
License window	The time period in which the MSO is allowed to show to its customers
Metadata	The generic term for all sorts of captured data that relates in one way or another to essence material; media related metadata (SD/HD, 4:3 vs 16:9, etc), title metadata (title, description, directors etc) and terms metadata (e.g. rental period).
Poster	Image asset to be shown in the catalogue.
Preview/trailer	Asset which is the item that will (often) be delivered free of charge as an advance viewing of several scenes advertising a forthcoming movie.
Price promotion	A temporarily price reduction of the product for promotional purposes
PRODIS	An SeaChange system supporting the management of assets and products
Product	A commercial offer, to be sold to consumers, consisting of one or more assets
Title	An identifying name given to an asset
TRAXIS	An SeaChange system supporting processes involved in VOD transactions
Rental period	The time period following the actual product purchase by the consumer for which the product may be viewed before a new purchase would be required.
Viewing period	The time period in which the customer is allowed to view the asset.

1.5 Conventions

The following conventions are applicable in this document:

- The word *shall* is used to indicate mandatory requirements strictly to be followed and from which no deviation is permitted *(shall* equals *is required to)*.

- The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited *(should* equals *is recommended that)*.
- The word *must* is used only to describe unavoidable situations.
- The word *will* is only used in statements of fact.
- The word *may* is used to indicate a course of action permissible within certain limits *(may* equals *is permitted to)*.
- The word *can* is used for statements of possibility and capability, whether material, physical, or causal *(can* equals *is able to)*.

1.6 Standards

Date time formats between the systems are ALWAYS in UTC time and W3C (ISO 8601 profile) formatting, e.g.: *2004-11-05T13:15:30Z*, unless specifically stated otherwise.

This way time discontinuities can be avoided (i.e. daylight savings). Note that all interfacing systems must decode/encode the date time to the correct local time.

1.7 References

This section lists the references made in this document.

[REST] Representational State Transfer – Architectural Styles and the Design of Network-based Software Architectures – Roy Fielding
http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

1.8 Open Issues

2 Introduction

The content propagation interface specifies the command interface used to manage content that resides on a video server.

The interface has the following tasks:

1. Copy content from a storage location, such as ftp or unc server into the video server.
2. Capture live content presented to a video server via a unicast or multicast UDP stream
3. Remove content from the video server storage device
4. Retrieve a list of content on the video server storage device
5. Retrieve detailed content information, such as bitrate and status, from the video server storage device

A video server using this interface may implement the mentioned transfer protocols only partially, e.g. only ftp transfers. A video server may also ignore part of the content attributes, such as name without affecting the usefulness of the interface.

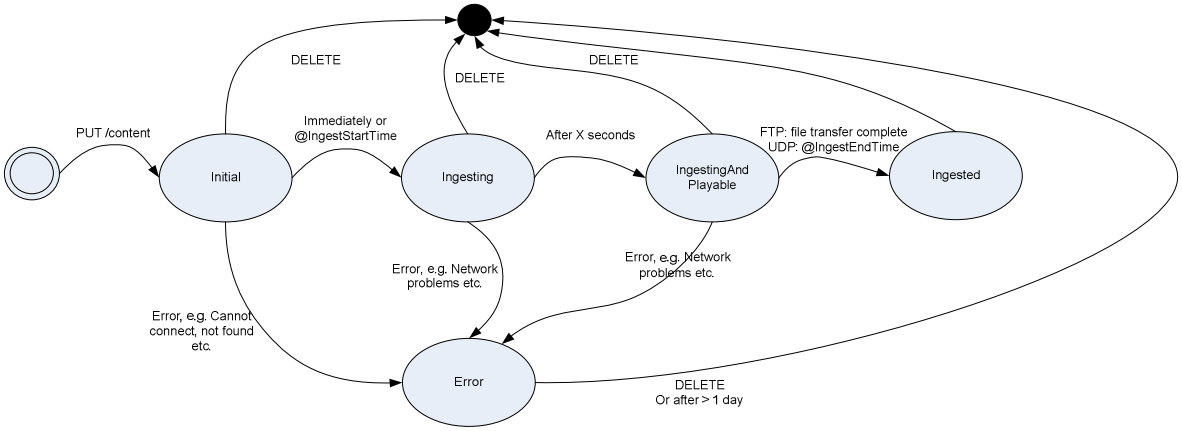
The ingest interface is asynchronous, unless mentioned otherwise. A PUT request to schedule a new ingest will complete directly, whereas the ingest itself may take minutes or hours.

3 Content propagation interface

The content propagation interface specifies how to manage content on a video server storage device. This chapter describes the exact structure of the interface.

3.1 State diagram

The content interface has the following state diagram.



When the content definition is PUT into the video server by the client, the content gets a state ‘Initial’. When the content starts ingesting at the pre-defined time or (when no ingest start time is provided) immediately, the state becomes ‘Ingesting’. At some point the amount of bytes in the video server buffer is sufficient to allow streaming to a set-top box and the state becomes ‘IngestingAndPlayable’. When the ingest finishes, the content moves into a state ‘Ingested’.

At each of the states, and error condition may happen that causes the content to assume the Error state. This state can only be resolved by DELETE-ing and PUT-ting the content again. In case the client forgets, the video server is allowed to delete content that is in ‘Error’ state from the system after at least one day.

3.2 Interface binding

The interface binding is based on REST services using HTTP according Appendix C: RESTful services.

An XML structure is optionally used in the standard body of the HTTP PUT or POST requests. Per call the XSD type of the XML structure is mentioned. The xsd is included as appendix. The DELETE request never contains an XML structure. On each type of HTTP request an XML structure in the response body can be returned. The response body of the HTTP GET request will always contain an XML structure.

The following semantics are applicable for the interface protocol:

- HTTP/1.1
- Application/xml as media type

3.2.1 Response handling

The response on a HTTP request is an HTTP status codes according to the “Success” or “Client Error” classes (i.e. status codes 2xx, 4xx or 5xx) of status codes.

The following HTTP status codes in a HTTP response are used in the online reporting interface:

Code	Definition	Description
200	OK	The request has been successfully executed and optionally an XML structure is returned with the requested information
400	Bad Request	The request contains invalid syntax or cannot be fulfilled
403	Forbidden	The server understood the request but is refusing to fulfil it.
404	Not Found	The resource of the request does not exist in the database
500	Internal Server Error	The request caused an internal server error

3.3 Resources

The following resources are used in the online content propagation REST interface:

- Content

3.4 Resource properties

The following complex XML types are returned by the REST interface. The XML schemas of the HTTP operations are included in Appendix B: REST services XML schemas.

3.4.1 Content

Properties

Content is represented by the *ContentType* complex type. *ContentType* contains the information of the VOD content on a video server.

The following figure presents the elements and attributes of the *ContentType*.

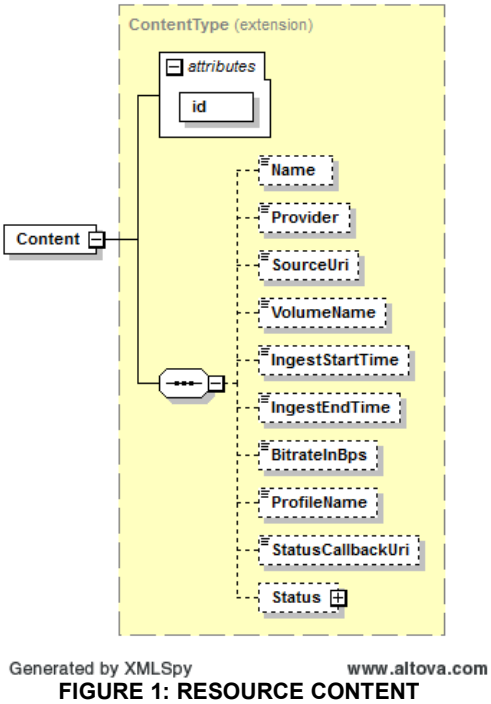


FIGURE 1: RESOURCE CONTENT

Field	Description	Req.
Element: Name	The content name. Maximum of 50 characters.	No
Element: Provider	The name of the content provider. Maximum of 50 characters.	No
Element: SourceUri	The location where the content on the video server has been copied from or should be copied from. Typical transfer protocols implemented on video servers are FTP, UNC (file transfer). For live capturing different methods can be used such as RTSP and UDP streaming or Channel aliases (channel mapping's) for OTT channels.	Yes*
Element: VolumeName	A free-format name where the content is stored on the video server for this client. May be a RAID-array id, a volume, a directory, a symbolic link name, etc. Server vendor shall provide a value to be used in a deployment.	No
Element: IngestStartTime	Start time of the ingest. When omitted in a PUT the content starts ingesting immediately. Typically used when recording live content via the location mentioned in the SourceUri, but may also be provided for file transfer protocols such as ftp. In that case it describes when the FTP transfer shall commence or has commenced.	No
Element: IngestEndTime	End time of the ingest. May only be used in a PUT statement when SourceUri is a capturing/recording location. When provided, endTime shall always be later than StartTime. The value may be filled for GET in file transfer scenarios such as FTP. In this case the field indicates when the video server finished ingesting the content.	No
Element: BitrateInBps	The content bitrate in bit per second. The bitrate shall be provided for live recordings via UDP in order for the video server to do resource allocation. The bitrate is not mandatory for file transfers such as FTP. The video server shall calculate the exact maximum bitrate upon ingest. Any subsequent requests for content bitrate will return the calculated bitrate.	No
Element: ProfileName	Indication of the physical content type in multiscreen scenarios.	No
Element: StatusCallbackUri	A URL used by the video server to send status callbacks for this ingest process. If specified, the video server shall use PUT operations to notify the back-office of state changes during ingest.	No
Element: Status	The enclosing element for content status information. The content status can never be set in a PUT operation, only retrieved via a GET operation.	No
Attribute: id	The status id shall be either one of the following values: • Initial: Content ingest has not yet started. • Ingesting: The content is being ingested • IngestingAndPlayable: The content is still being ingested but can already be used for playback. • Ingested: The content was ingested successfully and is playable. • Error-OutOfStorageSpace: The video server has no more storage space available. • Error-InsufficientIngestBandwidth: The video server has no more ingest capacity available. • Error-IngestHostUnreachable: The host referred to in the	Yes

Field	Description	Req.
	SourceUri is not reachable. • Error-IngestSourceNotFound: The content cannot be found on the location described by the SourceUri • Error-Other: Generic error message. The ErrorMessage element should contain more information. If the ingest was not successful the client may retry by means of a DELETE and a new PUT of the content properties. If the client chooses not to retry the content ingest this should be notified explicitly by using an HTTP DELETE. In case of an error the video server is allowed to issue an internal DELETE after at least one day to avoid dangling resources.	
Element: IngestPercentage	An integer value between 0 and 100 that indicates the progress of the ingest. The value is only required when the status id = 'Ingesting'. The value may also be returned with other status id values, such as an error to indicate at what point of the ingest the error occurred.	No
Element: ErrorMessage	A more descriptive error message, mostly used for 'Error-Other'.	No
Element: ErrorTimestamp	The timestamp at which the error occurred. Only used for 'Error' states.	No
Attribute: id	identification of the content	Yes

* Not required when only the status is retrieved.

URI template

/Content/{ContentId}

HTTP operation	Request body	Response body
PUT	Content object	N/A
GET	N/A	Content object
DELETE	N/A	N/A

Response status codes

The following response status codes can be returned for the HTTP operations:

200	PUT	The properties of a content are updated
	GET	The properties of a content are returned
	DELETE	The content and its properties are deleted from the video server
201	PUT	A new content ingest request with properties is added
400	PUT, GET, DELETE	The request contains invalid syntax or cannot be fulfilled.
403	PUT, DELETE	The content cannot be updated or deleted, e.g. because the content is still being streamed.
404	GET	The content does not exist in the video server content storage

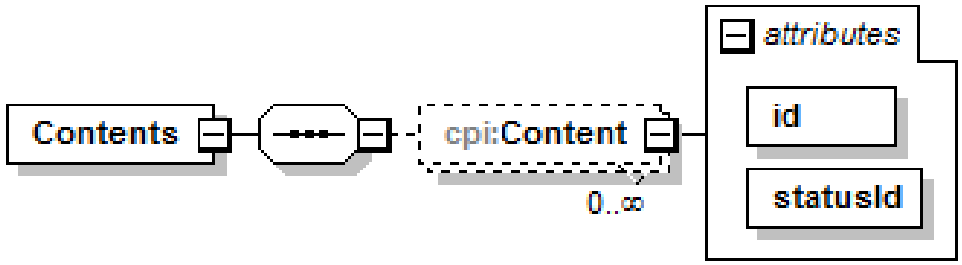
Remark

The status property may also be requested (GET) separately, since this will be the most accessed value:

/Content/{ContentId}/Status

3.4.2 Contents

Properties



Generated by XMLSpy www.altova.com

Field	Description	Req.
Element: Contents	The enclosing element for an enumeration of all content ids currently present on the video server storage device.	Yes
Element: Content	The enclosing element for the content id and its status	No
Attribute: id	The unique id of the content. The id can be used to retrieve more information via the /Content/{id} method.	Yes
Attribute: statusId	The content status as represented by an enumeration. The values are described in more detail in section 3.4.1..	Yes

URI template

/Contents

/Contents?Status=Ingesting

HTTP operation	Request body	Response body
GET	N/A	Contents object

Response status codes

The following response status codes can be returned for the HTTP operations:

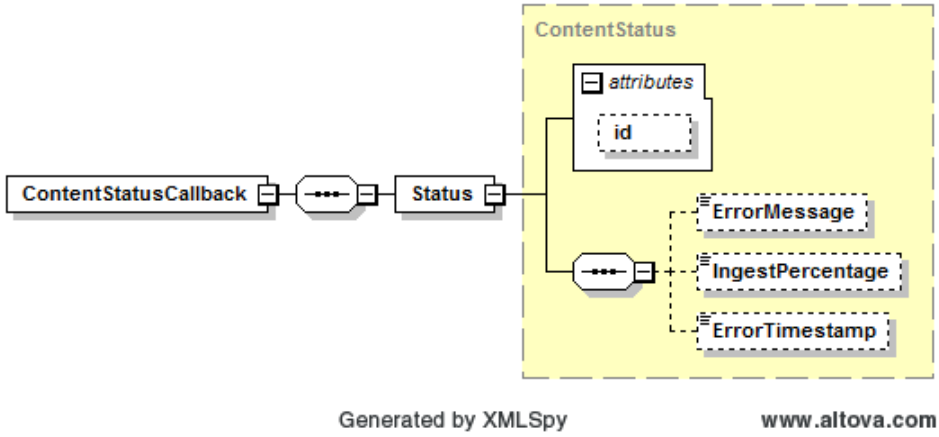
200	GET	All content information matching the criteria is returned in the body.
400	GET	The request cannot be fulfilled

Remark

The Status filter must be one of the StatusType fields as described in section 3.4.1.

3.4.3 ContentStatusCallback

Properties



Field	Description	Req.
Element: ContentStatusCallback	The enclosing element for a content status callback	Yes
Element: Content	The enclosing element for the content id and its status	Yes
Attribute: id	The status id shall be either one of the following values: • Initial: Content ingest has not yet started. • Ingesting: The content is being ingested • IngestingAndPlayable: The content is still being ingested but can already be used for playback. • Ingested: The content was ingested successfully and is playable. • Error-OutOfStorageSpace: The video server has no more storage space available. • Error-InsufficientIngestBandwidth: The video server has no more ingest capacity available. • Error-IngestHostUnreachable: The host referred to in the SourceUri is not reachable. • Error-IngestSourceNotFound: The content cannot be found on the location described by the SourceUri • Error-Other: Generic error message. The ErrorMessage element should contain more information. If the ingest was not successful the client may retry by means of a DELETE and a new PUT of the content properties. If the client chooses not to retry the content ingest this should be notified explicitly by using an HTTP DELETE. In case of an error the video server is allowed to issue an internal DELETE after at least one day to avoid dangling resources.	Yes
Element: IngestPercentage	An integer value between 0 and 100 that indicates the progress of the ingest. The value is only required when the status id = 'Ingesting'. The value may also be returned with other status id values, such as an error to indicate at what point of the ingest the error occurred.	No

Element: ErrorMessage	A more descriptive error message, mostly used for 'Error-Other'.	No
Element: ErrorTimestamp	The timestamp at which the error occurred. Only used for 'Error' states.	No
Element: IngestPercentage	An integer value between 0 and 100 that indicates the progress of the ingest. The value is only required when the status id = 'Ingesting'. The value may also be returned with other status id values, such as an error to indicate at what point of the ingest the error occurred.	

HTTP operation	Request body	Response body
PUT	ContentStatusCallback	N/A

Response status codes

The following response status codes can be returned for the HTTP operations:

200	PUT	The notification has been delivered successfully.
404	PUT	Not found. The content ingest task does not exist.
500	PUT	A server error has occurred.

Remark

The video server should not attempt to re-deliver notifications that fail.

3.5 Usage

Appendix A: Usage contains several usage scenarios of the content propagation interface.

4 Appendix A: Usage content propagation interface

This appendix contains some examples how the content propagation interface is used.

4.1 Generic error

```
HTTP/1.1 404 Not Found
```

4.2 Content operations

4.2.1 Start new FTP ingest immediately

```
PUT /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/ HTTP/1.1

<?xml version="1.0" encoding="utf-8" ?>
<Content id="4d854d85-4e6c-d06f-b15b-d06fa6d85edf" xmlns="urn:eventis:cpi:1.0">
  <Name>Spiderman 3</Name>
  <Provider>Sony</Provider>
  <SourceUri>ftp://10.0.0.10/Spiderman3.mpg</SourceUri>
  <ProfileName>MPEG2_SD</ProfileName>
</Content>

HTTP/1.1 200 OK
```

4.2.2 Start new FTP ingest at 21:00h

```
PUT /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/ HTTP/1.1

<?xml version="1.0" encoding="utf-8" ?>
<Content id="4d854d85-4e6c-d06f-b15b-d06fa6d85edf" xmlns="urn:eventis:cpi:1.0">
  <Name>Spiderman 3</Name>
  <Provider>Sony</Provider>
  <SourceUri>ftp://10.0.0.10/Spiderman3.mpg</SourceUri>
  <IngestStartTime>2009-09-13T21:00:00Z</IngestStartTime>
  <ProfileName>MPEG2_SD</ProfileName>
</Content>

HTTP/1.1 200 OK
```

4.2.3 Start new UDP live ingest for event from 21:00h to 22:00h

Note that the bitrate is mandatory for live ingests in order for the resource management to be able to schedule ingests over the available ingest ports.

```
PUT /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/ HTTP/1.1

<?xml version="1.0" encoding="utf-8" ?>
<Content id="4d854d85-4e6c-d06f-b15b-d06fa6d85edf" xmlns="urn:eventis:cpi:1.0">
  <Name>The nine o'clock news</Name>
  <Provider>BBC</Provider>
```

```
  <SourceUri>udp://224.0.0.1</SourceUri>
  <IngestStartTime>2009-09-13T21:00:00Z</IngestStartTime>
  <IngestEndTime>2009-09-13T22:00:00Z</IngestEndTime>
  <BitrateInBps>4500000</BitrateInBps>
</Content>

HTTP/1.1 200 OK
```

4.2.4 Delete content (independent of status)

Note: Any running streams will have been actively torn down by the SeaChange TRAXIS.SRM system before this command is issued.

After the call returns successfully the client may assume the content is no longer present on the video server. It may re-ingest content with the same id.

```
DELETE /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/ HTTP/1.1

HTTP/1.1 200 OK
```

It is possible the content cannot be removed, e.g. when not all streams have been torn down. It is the responsibility of the client to retry the delete operation later.

```
DELETE /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/ HTTP/1.1

HTTP/1.1 403 Forbidden
```

4.2.5 Stop scheduled or running FTP, UDP ingest

This scenario is identical to deleting the content, including the scenario that a 403 is returned if streams are still running.

```
DELETE /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/ HTTP/1.1

HTTP/1.1 200 OK
```

4.2.6 Update content - Delay start time of ingest from 21:00h to 21:15h

Typically used for live recording when the event that needs to be recorded is delayed. Note that an update must contain the full Content metadata.

```
PUT /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/ HTTP/1.1

<?xml version="1.0" encoding="utf-8" ?>
<Content id="4d854d85-4e6c-d06f-b15b-d06fa6d85edf" xmlns="urn:eventis:cpi:1.0">
  <Name>The nine o'clock news</Name>
  <Provider>BBC</Provider>
  <SourceUri>udp://224.0.0.1</SourceUri>
  <IngestStartTime>2009-09-13T21:15:00Z</IngestStartTime>
  <IngestEndTime>2009-09-13T22:00:00Z</IngestEndTime>
  <BitrateInBps>4500000</BitrateInBps>
</Content>
```

```
HTTP/1.1 200 OK
```

4.2.7 Update content - Delay start time but ingest has already commenced

The ingest has already started so the start cannot be delayed anymore. An error is returned and the status of the content and ingest is unchanged.

```
PUT /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/ HTTP/1.1

<?xml version="1.0" encoding="utf-8" ?>
<Content id="4d854d85-4e6c-d06f-b15b-d06fa6d85edf" xmlns="urn:eventis:cpi:1.0">
  <Name>The nine o'clock news</Name>
  <Provider>BBC</Provider>
  <SourceUri>udp://224.0.0.1</SourceUri>
  <IngestStartTime>2009-09-13T21:15:00Z</IngestStartTime>
  <IngestEndTime>2009-09-13T22:00:00Z</IngestEndTime>
  <BitrateInBps>4500000</BitrateInBps>
</Content>

HTTP/1.1 403 Forbidden
```

4.2.8 Update content - Delay end time of ingest from 22:00h to 22:15h

Typically used for live recording when the event that needs to be recorded is extended. This example assumes the original IngestEndTime is not in the past.

```
PUT /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/ HTTP/1.1

<?xml version="1.0" encoding="utf-8" ?>
<Content id="4d854d85-4e6c-d06f-b15b-d06fa6d85edf" xmlns="urn:eventis:cpi:1.0">
  <Name>The nine o'clock news</Name>
  <Provider>BBC</Provider>
  <SourceUri>udp://224.0.0.1</SourceUri>
  <IngestStartTime>2009-09-13T21:00:00Z</IngestStartTime>
  <IngestEndTime>2009-09-13T22:15:00Z</IngestEndTime>
  <BitrateInBps>4500000</BitrateInBps>
</Content>

HTTP/1.1 200 OK
```

4.2.9 Update content - Delay end time but ingest has already completed

The ingest has already finished so the end time cannot be delayed anymore. An error is returned and the status of the content and ingest is unchanged.

```
PUT /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/ HTTP/1.1

<?xml version="1.0" encoding="utf-8" ?>
<Content id="4d854d85-4e6c-d06f-b15b-d06fa6d85edf" xmlns="urn:eventis:cpi:1.0">
  <Name>The nine o'clock news</Name>
  <Provider>BBC</Provider>
```

```
  <SourceUri>udp://224.0.0.1</SourceUri>
  <IngestStartTime>2009-09-13T21:00:00Z</IngestStartTime>
  <IngestEndTime>2009-09-13T22:15:00Z</IngestEndTime>
  <BitrateInBps>4500000</BitrateInBps>
</Content>

HTTP/1.1 403 Forbidden
```

4.2.10 Update content – New SourceUri is provided

Ingesting from a new location requires a DELETE and a PUT. Trying to update the SourceUri will result in an error.

```
PUT /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/ HTTP/1.1

<?xml version="1.0" encoding="utf-8" ?>
<Content id="4d854d85-4e6c-d06f-b15b-d06fa6d85edf" xmlns="urn:eventis:cpi:1.0">
  <Name>Spiderman 3</Name>
  <Provider>Sony</Provider>
  <SourceUri>ftp://10.0.0.20/Spiderman3.mpg</SourceUri>
  <IngestStartTime>2009-09-13T21:00:00Z</IngestStartTime>
</Content>

HTTP/1.1 403 Forbidden
```

4.2.11 Get details for content element

```
GET /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/ HTTP/1.1

HTTP/1.1 200 OK

<?xml version="1.0" encoding="utf-8" ?>
<Content id="4d854d85-4e6c-d06f-b15b-d06fa6d85edf" xmlns="urn:eventis:cpi:1.0">
  <Name>Spiderman 3</Name>
  <Provider>Sony</Provider>
  <SourceUri>ftp://10.0.0.10/Spiderman3.mpg</SourceUri>
  <IngestStartTime>2009-09-13T13:11:00Z</IngestStartTime>
  <IngestEndTime>2009-09-13T13:13:00Z</IngestEndTime>
  <BitrateInBps>3750000</BitrateInBps>
  <Status id=”Ingested”/>
</Content>
```

4.2.12 Get details for unknown content element

```
GET /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/ HTTP/1.1

HTTP/1.1 404 Not found
```

4.2.13 Get status only of content element

It is allowed to retrieve the status property separately of the other properties.

```
GET /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/Status HTTP/1.1

HTTP/1.1 200 OK

<?xml version="1.0" encoding="utf-8" ?>
<Content id="4d854d85-4e6c-d06f-b15b-d06fa6d85edf" xmlns="urn:eventis:cpi:1.0">
  <Status id="Ingesting">
    <IngestPercentage>20</IngestPercentage>
  </Status>
</Content>
```

4.2.14 Get status only of content element in error

It is allowed to retrieve the status property separately of the other properties. This example shows an example of an out of storage space (typically disk space) error. Note that any residual content may still reside on the video server and should be cleaned with a DELETE statement. The video server is allowed to self-clean but not before 24 hours after the error occurred.

```
GET /Content/4d854d85-4e6c-d06f-b15b-d06fa6d85edf/Status HTTP/1.1

HTTP/1.1 200 OK

<?xml version="1.0" encoding="utf-8" ?>
<Content id="4d854d85-4e6c-d06f-b15b-d06fa6d85edf" xmlns="urn:eventis:cpi:1.0">
  <Status id="Error-Other">
    <ErrorMessage>Video server shutting down</ErrorMessage>
    <IngestPercentage>30</IngestPercentage>
    <ErrorTimestamp>2009-09-08T12:00:34Z</ErrorTimestamp>
  </Status>
</Content>
```

4.2.15 Get all content in ‘ingesting’ state

This method returns all content that is currently being ingested.

```
GET /Contents?Status=Ingesting HTTP/1.1

HTTP/1.1 200 OK

<?xml version="1.0" encoding="utf-8" ?>
<Contents xmlns="urn:eventis:cpi:1.0">
  <Content id="4d854d85-4e6c-d06f-b15b-d06fa6d85edf" statusId="Ingesting"/>
  <Content id="34854e55-1e7d-e06a-a15a-a06fa6a55fff" statusId="Ingesting"/>
</Contents>
```

5 Appendix B: REST services XML schemas

The following XSD describes the content propagation XML schema:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright (C) 2009 SeaChange Interactive Solutions BV. All rights reserved. -->
<schema xmlns:cpi="urn:eventis:cpi:1.0" attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="urn:eventis:cpi:1.0" xmlns="http://www.w3.org/2001/XMLSchema">
  <annotation>
    <documentation>eventIS content propagation interface XML schema</documentation>
  </annotation>
  <simpleType name="StatusType">
    <restriction base="string">
      <enumeration value="Initial" />
      <enumeration value="Ingesting" />
      <enumeration value="IngestingAndPlayable" />
      <enumeration value="Ingested" />
      <enumeration value="Deleting" />
      <enumeration value="Error-OutOfStorageSpace" />
      <enumeration value="Error-InsufficientIngestBandwidth" />
      <enumeration value="Error-IngestHostUnreachable" />
      <enumeration value="Error-IngestSourceNotFound" />
      <enumeration value="Error-Other" />
    </restriction>
  </simpleType>
  <complexType name="ContentType">
    <sequence>
      <element minOccurs="0" maxOccurs="1" name="Name" type="string" />
      <element minOccurs="0" maxOccurs="1" name="Provider" type="string" />
      <element minOccurs="0" maxOccurs="1" name="SourceUri" type="string" />
      <element minOccurs="0" maxOccurs="1" name="VolumeName" type="string" />
      <element minOccurs="0" maxOccurs="1" name="IngestStartTime" type="dateTime" />
      <element minOccurs="0" maxOccurs="1" name="IngestEndTime" type="dateTime" />
      <element minOccurs="0" maxOccurs="1" name="BitrateInBps" type="long" />
      <element minOccurs="0" maxOccurs="1" name="ProfileName" type="string" />
      <element minOccurs="0" maxOccurs="1" name="Status">
        <complexType>
          <sequence>
            <element minOccurs="0" maxOccurs="1" name="ErrorMessage" type="string" />
            <element minOccurs="0" maxOccurs="1" name="IngestPercentage" type="int" />
            <element minOccurs="0" maxOccurs="1" name="ErrorTimestamp" type="dateTime" />
          </sequence>
          <attribute name="id" type="cpi:StatusType" />
        </complexType>
      </element>
    </sequence>
    <attribute name="id" type="string" use="required" />
  </complexType>
  <element name="Content">
    <complexType>
      <complexContent mixed="false">
        <extension base="cpi:ContentType" />
      </complexContent>
    </complexType>
  </element>
  <element name="Contents">
    <complexType>
      <sequence>
        <element minOccurs="0" maxOccurs="unbounded" name="Content">
          <complexType>
            <attribute name="id" type="string" use="required" />
            <attribute name="statusId" type="cpi:StatusType" use="required" />
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>
</schema>
```

6 Appendix C: RESTful services

6.1 Background

Nowadays many systems use XML-based messages to exchange information over computer networks. Several protocols are specified in the past by several organizations to support the exchange of these XML-based messages over HTTP. Some of these protocols are for instance XML-RPC and SOAP. These protocols claim to offer Web services over HTTP. However, in many ways these protocols don't follow the architecture of the Web. SOAP is used for an interoperable cross-platform remote procedure call (RPC) but can be complex to handle between different platforms.

6.2 REST

Now there is REST. REST is an architectural style that uses the strengths of the Web to build services (more detailed information about REST can be found at [REST][REST]). It proposes a set of constraints that simplifies interfaces for integration. Interfaces that are compliant to REST are called RESTful services. REST embraces the Web where each resource is identified by a single URI. RESTful services model the interaction with user agents based on resources. Each such resource is represented by a unique URI, and the user agent uses the uniform interface of HTTP to interact with a resource via that URI.

User agents only interact with resources using the prescribed HTTP verbs. The four main verbs of the uniform HTTP interface are POST, GET, PUT, and DELETE. These verbs are often associated with the CREATE, READ, UPDATE and DELETE (CRUD) operations used in database technologies. The following table provides an overview of these commands:

HTTP	CRUD	Usage
POST	Create	Used to create a new resource in a collection where the host creates the resource (e.g. the user agent does not supply the unique resource). As a result this resource is returned to the user agent.
GET	Read	Used to retrieve a read-only representation of a resource
PUT	Update, Create	Used to modifying an existing resource in the host. In case the provided resource does not exist it will be created in the host (e.g. the user agent supplies the unique resource)
DELETE	Delete	Used to delete a resource in the host

In case of complex operations an XML structure can be used in the request and/or response to provide/retrieve attributes of the unique resource.

As a result of these HTTP commands a response is returned to the user agent using the standard HTTP response status codes.