Standard Transcoding Interface Specification

Candidate Version 1.0 – 07 Jun 2005

Open Mobile Alliance

OMA-TS-STI-V1_0-20050607-C

Contents

Figures

Tables

1. Scope

This document presents the normative and informative elements of a transcoding interface between a Transcoding Platform and Application Platforms. The primary purpose of the interface is to request transcoding of media Content files based on specified transcoding parameters, User Equipment capabilities and application policies and to return the corresponding transcoded media files.

The notion of "Content Adaptation" or "Transcoding" in this document refers to the transformation and manipulation of Content (images, audio, video, text, presentation…etc.) to meet the desired targets (defined by the terminal capabilities and the application needs). Those adaptations include: media format transcoding, scaling, re-sampling, file size compression…etc.

The document describes a number of interface exchange examples and proposes syntax and semantics for the first OMA STI specification.

2. References

2.1 Normative References

[C.S0050-0 V1.0] "3GPP2 File Formats for Multimedia Services", Version 1.0, URL: http://www.3gpp2.org

[FOURCC] "Video Codec and Pixel Format Information", Video Codecs, URL: http://www.fourcc.org/

[HTTP] HTTP v1.1 – RFC 2616, URL: http://www.w3.org/Protocols/rfc2616/rfc2616.html

[IANA] "Internet Assigned Numbers Authority", MIME Media Types, URL: http://www.iana.org/assignments/media-types

[IOPPROC] "OMA Interoperability Policy and Process", Version 1.1, Open Mobile Alliance™, OMA-IOP-Process-V1_1, URL: http://www.openmobilealliance.org/

[ISO/IEC 10918-1] ISO/IEC 14496-10:2001: "Information technology – Digital compression and coding of continuous-tone still images - Requirements and guidelines", URL: http://www.iso.org.

[ISO/IEC 14496-10] ISO/IEC 14496-10:2001: "Information technology – Coding of audio-visual objects – Part 10: Advanced video coding", URL: http://www.iso.org

[ISO/IEC 14496-2] ISO/IEC 14496-2:2001: "Information technology – Coding of audio-visual objects – Part 2: Visual", URL: http://www.iso.org

[ISO/IEC 14496-3] ISO/IEC 14496-3:2001: "Information technology – Coding of audio-visual objects – Part 3: Audio", URL: http://www.iso.org

[MIME] MIME Part 1, Format of Internet Message Bodies – RFC 2045, URL: http://www.ietf.org/rfc/rfc2045.txt

MIME Part 2, Media Types – RFC 2046, URL: http://www.ietf.org/rfc/rfc2046.txt

MIME Part 3, Message Header Extensions for Non-ASCII-Text – RFC 2047, URL: http://www.ietf.org/rfc/rfc2047.txt

MIME Part 4, Registration Procedures – RFC 2048, URL: http://www.ietf.org/rfc/rfc2048.txt

MIME Part 5, Conformance Criteria and Examples – RFC 2049, URL: http://www.ietf.org/rfc/rfc2049.txt

MIME Multipart/Related Content-Type – RFC 2387, URL: http://www.ietf.org/rfc/rfc2387.txt

[OMA MWS Guidelines] "OMA Web Services Enabler (OWSER): Core Specifications", Version 1.0, Open Mobile Alliance™, OMA-OWSER-Core-Specification-V1_0, URL: http://www.openmobilealliance.org

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels". S. Bradner. March 1997. URL: http://www.ietf.org/rfc/rfc2119.txt

[RFC2392] "Content-ID and Message-ID Uniform Resource Locators", E. Levinson, August 1998, URL: http://www.ietf.org/rfc/rfc2392.txt

[RFC3003] "The audio/mpeg Media Type", M. Nilsson, November 2000, URL: http://www.ietf.org/rfc/rfc3003.txt

[STI XSD] STI 1.0 XSD schema, Open Mobile Alliance™, OMA-SCHEMA_STI-V1_0, URL: http://www.openmobilealliance.org

[UAProf] "User Agent Profile", Version 2.0, Open Mobile Alliance™, OMA-UAProf-V2_0, URL: http://www.openmobilealliance.org

[WAPWSP] "OMA Wireless Session Protocol", Open Mobile Alliance™, WAP-203-WSP-20000504-a. URL: http://www.openmobilealliance.org.

[WS-I Attachment Profile 1.0] Web Services Interoperability, Attachment Profile, Version 1.0, URL: http://www.ws-i.org/Profiles/AttachmentsProfile-1.0-2004-06-11.html

[WS-I Basic Profile 1.1] Web Services Interoperability, Basic Profile 1.1,

URL: http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html

[WS-I SOAP Binding Profile 1.0] Web Services Interoperability, Simple SOAP Binding Profile, Version 1.0, URL: http://www.ws-i.org/Profiles/SimpleSoapBindingProfile-1.0-2004-08-24.html

[XML Schema Part 2: Datatypes] "XML Schema Part 2: Datatypes", W3C Recommendation 02 May 2001, URL: http://www.w3.org/

2.2 Informative References

[WS-I] Web Services Interoperability Organization, URL: http://www.ws-i.org/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

Application Platform	Combination of hardware and software that provide the functionality of an application. Note that rather than implementing all components to provide the functionality of an application the implementation can integrate the necessary components from other platforms.
Content	Subject matter or information that is processed, stored, or transmitted electronically. It includes such things as text, presentation, audio, images, video, etc. Content may have properties such as media type, mime type, etc.
Content Adaptation	The transformation and manipulation of Content (images, audio, video, text, presentation…etc.) to meet the desired targets (defined by the terminal capabilities and the application needs). These adaptations include: transforming media formats, scaling, re-sampling, file size compression…etc.
Job Result	Part of the Transcoding Response that corresponds to one individual transcoding, i.e. one transcoded content (which may contain one or more media elements) and the parameters corresponding to the particular transcoding as returned to the Application Platform
Multipart Content	A set of media elements that can be transcoded as a whole, in a single transcoding job
Policy	A set of rules that are specified by the Application Platform to the Transcoding Platform, that can be used to give general limitations and preferences as well as specific variations of the transcoding parameters up to the transcoding job granularity.
Profile	Set of parameters and constraints that define the transcoding target. Those parameters come from the User Equipment characteristics, combined with the specific application needs and describe the format, resolution, file size…etc. that the transcoded Content should conform to.
Request Body	The SOAP body of a Transcoding Request that contains the STI parameters of the request. A Request Body may contain one or more Transcoding Jobs.
Response Body	The SOAP body of a Transcoding Response that contains details about the performed transcoding. A Response Body may contain one or more Job Results.
Transcoding	Same as “Content Adaptation”, can be used interchangeably.
Transcoding Job	Part of the Transcoding Request that corresponds to one individual transcoding, i.e. one input content (which may contain more media elements in case of Multipart Content) and the parameters corresponding to the particular transcoding as given by the Application Platform.
Transcoding Platform	Combination of hardware and software that provide transcoding functionality.
Transcoding Request	A transcoding request, as issued by the Application Platform. The Transcoding Request can contain one or more Transcoding Job(s) and zero or more content attachment(s)
Transcoding Response	Transcoding Platform response that can contain the results of one or more Transcoding Job(s), and zero or more content attachment(s).
Transcoding Service	Functionality for Transcoding of Content offered as a service to an application.

User Equipment	A device allowing a user access to network services. For the purpose of OMA specifications the interface between the UE and the network is the radio interface

3.3 Abbreviations

13k	or QCELP or Q13: QualComm Code Excited Linear Predictive Coding at 13k
AAC	Advanced Audio Coding
AAC-LC	Advanced Audio Coding – Low Complexity
AMR	Adaptive Multi Rate
AMR-NB	Adaptive Multi Rate - Narrow Band
AMR-WB	Adaptive Multi Rate - Wide Band
ASCII	American Standard Code for Information Interchange
BMP	Bit Map
DRM	Digital Rights Management
EVRC	Enhanced Variable Rate Coder
FTP	File Transfer Protocol
GIF	Graphics Interchange Format
GIF 87a/89a	GIF with animations
HTTP	HyperText Transfer Protocol
JPEG	Joint Photographic Experts Group
JPEG-P	Joint Photographic Experts Group – Progressive
MIDI	Musical Instrument Digital Interface
MIME	Multipurpose Internet Mail Extension
MM	Multimedia Message
MMS	Multimedia Messaging Service
MMSC	Multimedia Messaging Service Center
MPEG	Moving Picture Experts Group
MP3	MPEG-1 Audio Layer 3
OMA	Open Mobile Alliance
PC	Personal Computer
PDA	Personal Digital Assistant
PNG	Portable Network Graphics
Q13	or 13k or QCELP: QualComm Code Excited Linear Predictive Coding at 13k
QCELP	or 13k or Q13: QualComm Code Excited Linear Predictive Coding at 13k
RFC	Request For Comments
SDP	Session Description Protocol
SMIL	Synchronized Multimedia Integration Language
SMV	Selectable Mode Vocoders
SOAP	Simple Object Access Protocol
SP-MIDI	Scalable Polyphony - Musical Instrument Digital Interface
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF	Unicode Translation Format
WAP	Wireless Application Protocol
WBMP	Wireless Bit Map
XML	eXtensible Mark-up Language

4. Introduction (Informative)

The deployment of multimedia applications (MMS, WAP (2), e-mail, web browsing… etc.) may require some Content Adaptations, due to the diversity of the phone specifications (memory, screen size, resolution, colour depth…etc. and supported media formats) and of the media formats as distributed by the Content industry (JPEG, GIF, all AMR modes, 13k vocoder, EVRC, MPEG-1, MPEG-4, H.263…etc.).

STI 1.0 is the first specification of a standard interface between Application Platforms and a Transcoding Platform and is meant to resolve some of the integration and testing problems when deploying multimedia services towards mobile devices.

This document describes a proposal for the standard interface, to be used for OMA STI 1.0.

5. Transcoding Interface (Normative)

5.1 Interface Overview

5.1.1 Architecture

Figure 1 provides a high level overview of the interaction between an Application Platform and a Transcoding Platform.

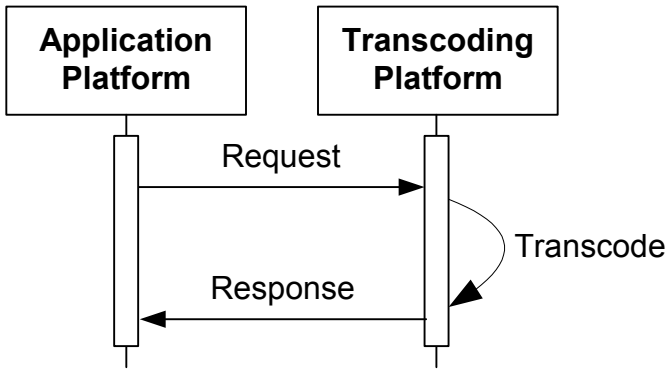

Figure 1Transcoding operation

The communication between the Application Platform and the Transcoding Platform SHALL be based on a request-response operation model, that is, a transcoding session is always a request followed by a response. A session is closed only when a response has been received. The Application Platform requests Transcoding from the Transcoding Platform. The Transcoding Platform receives the request, parses it, handles it and generates a response to the originating Application Platform.

Note:

This figure shows two separate logical entities – The Application Platform and the Transcoding Platform. These entities may share the same physical machine, or even be parts of the same software.

5.1.2 Protocol choice

The interface between the Application Platform and the Transcoding Platform SHALL rely on SOAP 1.1 over HTTP(S) (according to the SOAP guidelines as defined in the [OMA MWS Guidelines], section 6).

The SOAP request SHALL be sent as the body of a HTTP(S) POST request, as defined in SOAP 1.1 [OMA MWS Guidelines].

The SOAP response SHALL be sent as a body of the HTTP(S) response.

Both the request and the response SHALL contain a single SOAP envelope, conformant with SOAP 1.1 [OMA MWS Guidelines].

The attachment structure SHALL be conformant to the definition of SOAP with Attachments (see the SOAP with Attachments guidelines defined in the [OMA MWS Guidelines]) but the recommended ‘start parameter’ SHALL be compulsory, to permit more robust error detection.

5.1.3 Data types

The datatypes of the STI parameters used in this specification are defined in the section "Built-in datatypes" of [XML Schema Part 2: Datatypes] specification by the W3C, as adapted by SOAP 1.1 [OMA-MWS-Guidelines]. The datatypes discussed there are computer representations of well known abstract concepts such as integer and date.

Following are informative examples of built-in datatypes. For their full definition refer to [XML Schema Part 2: Datatypes]:

- string represents character strings in XML, i.e. the set of finite-length sequences of characters.
- token represents tokenized strings, i.e. strings that do not contain the line feed (#xA) nor tab (#x9) characters, that have no leading or trailing spaces (#x20) and that have no internal sequences of two or more spaces.

Wherever applicable, STI implementations SHALL use the canonical lexical representation of these datatypes, as defined in [XML Schema Part 2: Datatypes].

Following are informative examples of the canonical lexical representation of built-in datatypes. For their full definition refer to [XML Schema Part 2: Datatypes]:

- The canonical representation for integer is defined by prohibiting certain options from the Lexical representation. Specifically, the preceding optional "+" sign is prohibited and leading zeroes are prohibited: so '+00012' becomes '12'
- The canonical representation for unsignedLong is defined by prohibiting certain options from the Lexical representation. Specifically, leading zeroes are prohibited: so '0000321323' becomes '321323'

In addition to the XML datatypes, STI 1.0 also defines the following datatypes:

- unboundedLong represents all Long integer values that are greater than or equal to 0, where the value of infinity is lexically expressed as -1.
- unboundedInt represents all Int values that are greater than or equal to 0, where the value of infinity is lexically expressed as -1.
- nonNegativeInt represents all Int values that are greater than or equal to 0. This type is used instead of the standard XML "unsignedInt" which cannot be mapped with the same range of values to all programming languages.
- nonNegativeLong represents all Long values that are greater than or equal to 0. This type is used instead of the standard XML "unsignedLong" which cannot be mapped with the same range of values to all programming languages.

5.1.4 Case Sensitivity

There are a few categories of strings in the STI protocol. For each of these, the following table defines whether they are considered case sensitive or not, and provides relevant examples.

String Category	Case sensitivity	Valid case example	Invalid case example
XML elements names (XML tags)	Case sensitive. Enforceable by the XML schema.	<transcodingParams>	<TranscodingParams>
Values of XML enumeration (enumeration literals)	Case sensitive. Enforceable by the XML schema.	<layout>Portrait</layout>	<layout>portrait</layout>
Any other string	Case insensitive. Whenever applicable, implementations	Recommended: <contentType>image/gif</contentType>	N/A

	SHOULD use the same case as mentioned in the relevant tables of the spec. For values registered with [IANA] (or a different registration authority) and used in STI – implementation SHOULD use the same case as defined by the relevant registration authority.	Valid, but not recommended: <contentType>image/GIF</contentType>	
		Recommended: <transformation> <type>Mirror</type> <attributes> <property> <name>axis</name> <value>UD</value> </property> </attributes> </transformation>	
		Valid but **not** recommended: <transformation> <type>MIRROR</type> <attributes> <property> <name>Axis</name> <value>ud</value> </property> </attributes> </transformation>	

5.1.5 Transcoding Requests\Responses structure

5.1.5.1 Transcoding Request

Figure 2 shows an overview of the request components. Multiple bulk Transcoding within a single Transcoding Request is allowed by the proposed interface: therefore we will distinguish between Transcoding Job (individual media Transcoding) and Request Body, which can contain one or several Transcoding Jobs as part of one single request to the Transcoding Platform.

In case a Transcoding Request contains the Content files themselves, they SHALL be attached inside the HTTP(S) POST message, but outside the SOAP envelope.

The figure shows two configurations for a Transcoding Request within an HTTP POST Request. The first one is an HTTP request with a MIME Multipart body containing the SOAP Message as the first body part of the multipart and the content attachments as other parts of the body. The second is an HTTP request with a simple xml body which is the SOAP Message containing the STI Request Body. In this request there are no content attachments, but rather references to external content items.

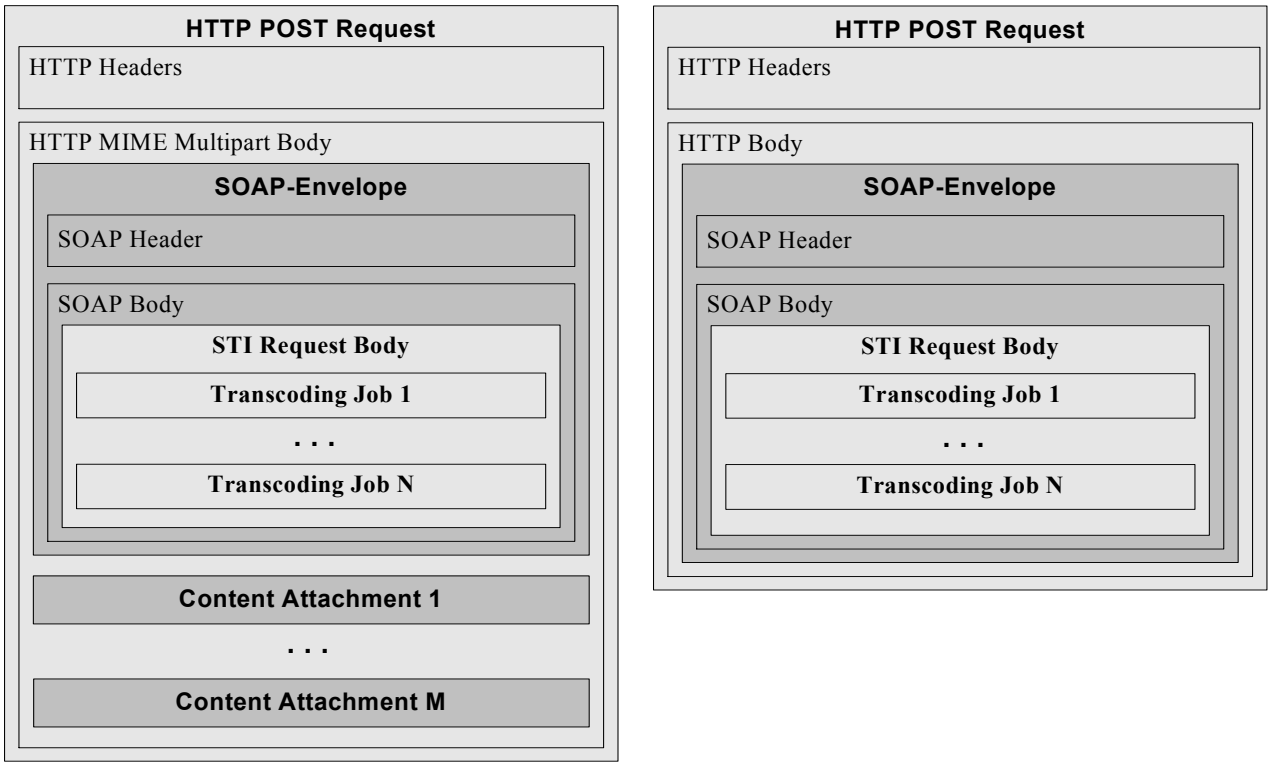

Figure 2Two configurations of a Transcoding Request structure within an HTTP POST Request

Note:

The SOAP Header (first part of the SOAP Envelope) is optional. STI 1.0 does not define any SOAP Header elements.

SOAP provides a flexible mechanism for extending a message in a decentralized and modular way without prior knowledge between the communicating parties. Typical examples of extensions that can be implemented as SOAP header entries are authentication, transaction management, payment etc. SOAP also defines a few attributes that can be used to indicate who should deal with a feature and whether it is optional or mandatory. Refer to the SOAP guidelines as defined in the [OMA MWS Guidelines], section 6 for more details.

Each Transcoding Job in the Request Body SHALL contain source parameters and MAY also contain target parameters, as presented in Figure 3.
For a complete list of the parameters of the Transcoding Job structure, please refer to section 5.2.2.

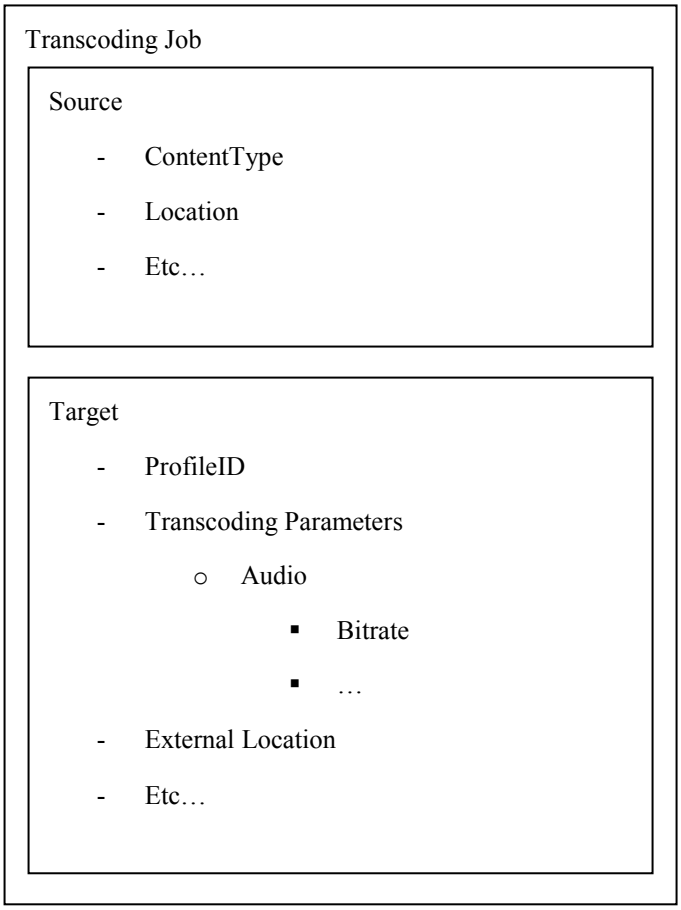

Figure 3Transcoding Job structure

5.1.5.2 Transcoding Response

The Transcoding Response (response of the Transcoding Platform to the Transcoding Request) contains the Jobs Results (the Transcoding Jobs' results). The Transcoding Response is detailed in Figure 4.

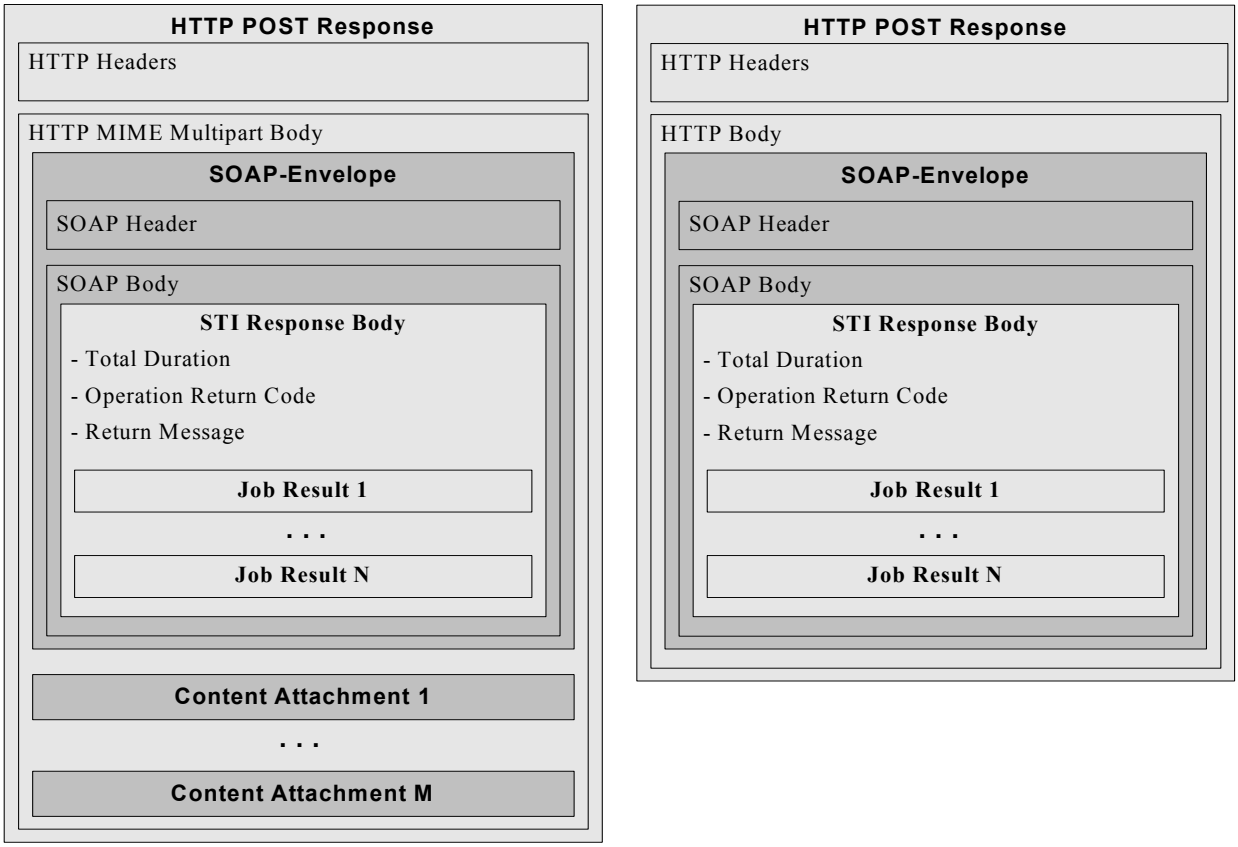

Figure 4Transcoding Response structure

In the Response Body, parameters either relate to the whole transcoding operation or the individual Transcoding Jobs' results. For example, the total duration parameter corresponds to the complete operation's duration. In the Job Results, there are parameters describing the particular Transcoding that was performed (e.g. the specific job's return code, return string, duration).

The proposed interface supports reporting of statistical data gathered during the Transcoding at the jobs' and the whole operation's levels.

The Job Results contained in the Transcoding Response are detailed in Figure 5.

For a complete list of the parameters of the Job Result structure, please refer to section 5.3.3

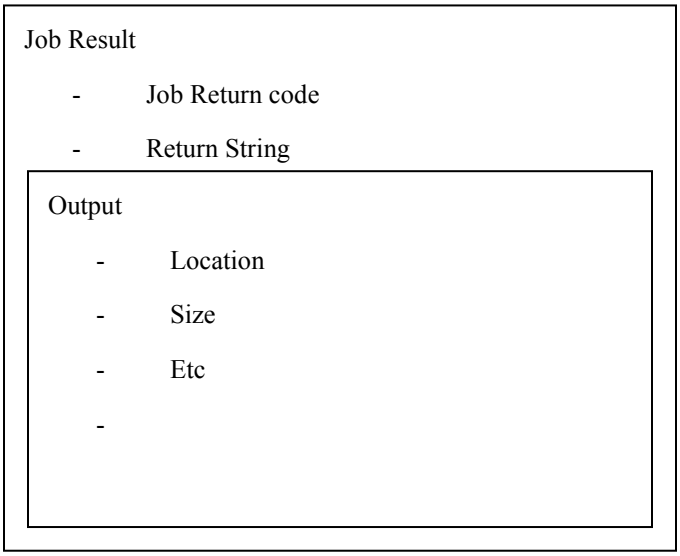

Figure 5Job Result structure

5.1.5.3 Source and Target Dependencies

The Source block of each Transcoding Job SHALL specify the source's content type and location. It MAY also specify additional parameters describing the source media.

If present, the Target block of each Transcoding Job MAY specify explicit transcoding parameters and/or Profile information and/or a URI for Policy parameters.

5.1.5.4 Support of Multipart Transcoding

The proposed interface also supports Transcoding Jobs having multipart Content, i.e. a set of media elements can be transcoded as a whole.

In this case, it is expected that the Transcoding Platform receives a multipart file, performs combined Transcoding of the different media elements (including any logical decisions between the different media elements) and recombines the elements into one transcoded multipart file as a response of the Transcoding Job.

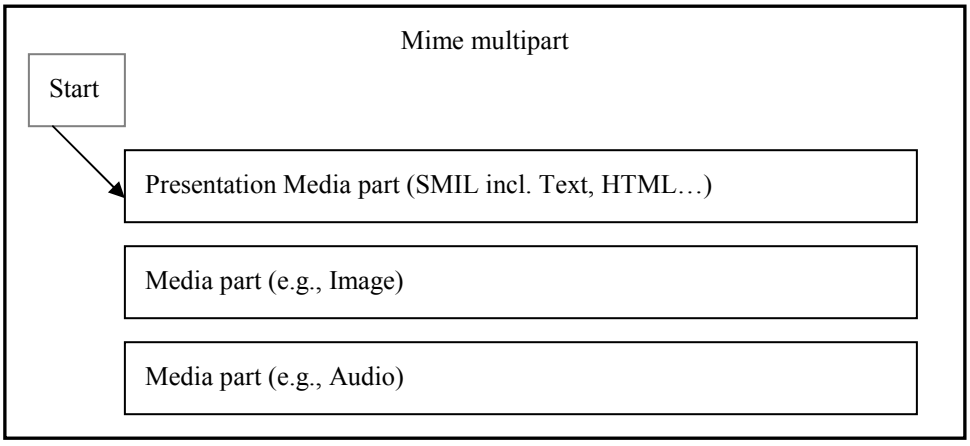

Figure 6Example of Multipart Format

The multipart (both input and output) MAY be transferred as a MIME-Multipart content item (see [MIME]), using the appropriate MIME types, as defined in [MIME] (e.g. multipart/related, multipart/mixed).

The multipart (both input and output) MAY also be transferred in the WAP binary form of the MIME Multipart using the appropriate MIME types, as defined in [WAPWSP] (e.g. application/vnd.wap.multipart.related, application/vnd.wap.multipart.mixed).

Figure *6* illustrates the structure of a multipart (with a presentation part).

5.1.6 Content Attachments

The Content data (either multipart or individual media parts) SHALL be referenced from within the SOAP Request Body (and Response Body), and SHALL reside either on a storage that can be accessed by the Transcoding Platform, or attached as part of the Transcoding Request/Response itself.

The proposed interface provides two methods of supporting media attachments:

- Self contained requests/responses, in which the Content data resides within the request/response itself
- References to external Content elements, in which case a Transcoding Job only contains a pointer to a remote location from where the Content elements can be pulled by the Transcoding Platform

The following two sections describe each of the methods. Note that the two options can be combined within a single operation: one Transcoding Request can contain job(s) having references (URIs) to external Content elements and job(s) having attached Content elements.

5.1.6.1 References to External Content Elements

In the case that references to external Content elements are used, the SOAP Request/Response Body SHALL contain the URIs pointing to the relevant files.

The Transcoding Platform SHALL support retrieval and upload of external content via HTTP(S). The Transcoding Platform MAY also support other protocols, for example, FILE and FTP (or any of its secured variants).

In case the required upload protocol (specified in the *externalLocation* parameter of the *target* element) is HTTP(S), the Transcoding Platform SHOULD use the HTTP(S) PUT method in order to upload the content.
Note that in this case, the HTTP server specified in the URL must also support the PUT upload method.

5.1.6.2 Self Contained Requests/Responses

In the case of self-contained requests/responses, the SOAP Request/Response Body SHALL contain references to the attachments.

The attachments SHALL be sent along with the Transcoding Request/Response, as MIME parts according to the SOAP with attachments definition (refer to the SOAP with Attachments guidelines as defined in the [OMA MWS Guidelines], section 6.8).

Each Content attachment header, both in the Transcoding Request and the Response, SHALL contain the unique ID (Content ID according to [RFC2392]) of the content, the content-type, and the content-transfer-encoding (e.g. base64, binary).

The SOAP Request/Response Body SHALL refer to the content, using the Content IDs. See section 5.2.1.2 for more details.

5.1.7 Transcoding Profiles, Parameters, Policies and Adaptation Classes

To specify the Transcoding Request details, the Application Platform SHALL use predefined Profile information, and/or explicit transcoding parameters, and MAY specify a URI for Policy parameters.

Upon receiving a Transcoding Request, the Transcoding Platform SHALL use the provided Profile information, explicit transcoding parameters, and Policy parameters in the order defined in this specification.

5.1.7.1 Using a Predefined Profile

The transcoding parameters MAY be indicated using the *profileID* parameter to reference a pre-defined Profile.

The Profile represents information to be considered when performing Transcoding. Often, the Profile will describe the User Equipment characteristics.

The Profile can be referenced using the User-Agent header, the UAProf string (URL) [UAProf], or a proprietary string. The User-Agent header is normally part of HTTP/WSP transactions between User Equipment and Application servers. This header is widely used. UAProf information is expressed as a link (URL) to the location of the terminal information details. The UAProf URL reference MAY be a reference to a UAProf database that both the Transcoding and Application Platforms share.

The Transcoding Platform SHOULD support usage of User Agent or UAProf URL as values of the *profileID* parameter.

The *profileID* parameter MAY be complemented by the *applicationType* parameter, in order to refer the Transcoding Platform to specific capabilities within the Profile (e.g. whether MMS capabilities or browsing capabilities of the UAProf are to be used)

The definition of a Profile and its content are out of the scope of this specification.

5.1.7.2 Using Explicit Transcoding Parameters

The transcoding parameters MAY also be indicated using an explicit list of parameters, which will reflect the characteristics of the target device and/or the specific requirements from the Application Platform.

When a Profile is indicated and explicit parameters are added, the explicit parameters SHALL override the corresponding parameters in the referenced Profile.

See section 5.2.6 for a list of all the supported transcoding parameters.

5.1.7.3 ProfileID and Transcoding Parameters Hierarchy

It is possible to specify a Profile (*profileID* parameter) and/or transcoding parameters (*transcodingParams* parameter) both in the request level and the job level.

If a *profileID* parameter is specified in the job level (lower level), then the *profileID* in the request level, if given, SHALL be ignored for this job.

If a *transcodingParams* parameter is specified in the job level (lower level), then the *transcodingParams* in the request level, if given, SHALL be ignored for this job.

If both a *profileID* parameter and a *transcodingParams* parameter exist, in either the request or the job level, the parameters in the *transcodingParams* SHALL take precedence over the corresponding parameters in the referenced profile.

In case the *transcodingParams* are not compatible with the capability profile (e.g. the content-type requested in the *transcodingParams* is not supported according to the Profile), the Transcoding Platform SHOULD perform the requested Transcoding but issue a warning in the response.

5.1.7.4 Using Policy Parameters

Policy parameters are a mean for the Application Platform to specify general rules for the Transcoding Request that define the behaviors of the Transcoding Platform (e.g. priority order between the different media elements, whether or not to use a default Profile, etc.).

Only a reference (URI) to external policies information is standardized in STI 1.0 (*policyRef* parameter). This means that a set of policies MAY be referenced in a Transcoding Request coming from an Application Platform. The Transcoding Platform SHOULD take the referenced policies into consideration.

The exact definition of Policies, their format and content are not in the scope of STI 1.0.

5.1.7.5 Using Adaptation Classes

There are situations where an Application Platform needs to describe to the Transcoding Platform certain classes of adaptations and specify if they are allowed or not. For instance, in the MMS application, the MMS proxy/relay may need to describe to the Transcoding Platform what constitutes "major" adaptations (in the context of MMS) and specify that they are not allowed. The MMS proxy/relay may also want to learn if minor adaptations were performed. STI provides a mechanism that allows the Application Platform to control which adaptation classes are allowed and which are not, and to find out from which adaptation classes were the adaptations that were performed.

The detailed description of the Adaptation Classes is not in the scope of STI 1.0.

The STI Transcoding Request contains parameters to specify which adaptation classes are of interest to the Application Platform and control which ones are allowed (e.g. MMS minor or major adaptations are of interest and major is not allowed).

The Transcoding Response contains a list of adaptation classes which are associated with the actual transcoding operation performed (e.g. if minor, major or both classes of adaptations have been performed).

A Transcoding Platform SHALL not perform any disallowed adaptation.

Examples of the usage of Adaptation Classes are presented in Appendix E.

5.1.7.6 Using Size Limit Parameters

It is possible to specify a size limit parameter at the following levels:

a) Transcoding Request (*sizeLimit* parameter in *transcodingParams* of *TranscodingRequest*)

b) Transcoding Job (*sizeLimit* parameter in *transcodingParams* of *transcodingJob*)

c) Media (*sizeLimit* parameter in the respective media structures)

d) Application (*applicationSizeLimit* parameter in *TranscodingRequest* and *Target*)

If present at the Transcoding Request level, it does not represent the sum of the sizes from all the Transcoding Jobs within the request; it represents a guideline to be followed separately by each Transcoding Job within this request. Having a *sizeLimit* at the Transcoding Request level avoids having to specify the parameter for each individual Transcoding Job within the request.

If there is one at the Transcoding Job level, it SHALL take precedence over the one (if any) specified at the Transcoding Request level. Having a *sizeLimit* at the Transcoding Job level avoids having to specify the parameter for each specific media type within the Transcoding Job.

If there is one at the media level, it SHALL take precedence over the one (if any) specified at the Transcoding Request level and/or the one (if any) specified at the Transcoding Job level. The *sizeLimit* at the media level is per element, not for the total allowed for all the elements of the same type.

The "resulting" sizeLimit will be derived from the following hierarchy:

1. *sizeLimit* at the Transcoding Request level

   Can be overridden by:

2. *sizeLimit* at the Transcoding Job level

   Can be overridden by:

3. *sizeLimit* at the media level

As for multipart elements, note that *sizeLimit* specified for the media elements within a multipart SHALL be complementary to the *sizeLimit* specified for the multipart. For example, it is possible to set the *sizeLimit* of a multipart to 100KB and limit any audio part of the multipart to 50KB. Following are more examples:

1. Multipart sizeLimit: 100KB

   Image sizeLimit: 10KB

   Audio sizeLimit: 30KB

   If there are 3 images and 2 audio files (for example), everything should fit in the 100KB, since the maximum allowed is 90KB (3 x 10KB + 2 x 30KB = 90KB)

2. Multipart sizeLimit: 50KB

   Image sizeLimit: 10KB

   Audio sizeLimit: 30KB

   If there are 3 images and 2 audio files (for example), the total size of all the combined elements may exceed the allowed size at the Multipart level after the initial Transcoding, so additional adaptation in the form of truncating, deleting, quality reduction, bit rate reduction, etc may have to be performed.

3. Multipart sizeLimit: 30KB

   Image sizeLimit: 10KB

   Audio sizeLimit: 50KB

   The allowed size at the Multipart level will take precedence and it will not be possible to have audio files of 50KB. Similar to example 2 above, some additional adaptation in the form of truncating, deleting, quality reduction, bit rate reduction, etc may have to be performed.

Finally, if the *applicationSizeLimit* is specified, it SHALL be complementary to the result from the above sizeLimit parameters so that the final size SHALL NOT exceed the minimum of the "resulting" sizeLimit and the *applicationSizeLimit*. For example, an operator may decide to never have messages bigger than 300KB go through his network, no matter what the terminal capabilities are and therefore set the *applicationSizeLimit* to 300KB. In such a case, even if the target terminal capabilities allow for media larger than that, the final size of the media will not exceed the *applicationSizeLimit* of 300KB.

Note that *applicationSizeLimit* can be specified both in the *TranscodingRequest* and the *Target* elements. If present in the *Target* element, it SHALL take precedence over the one (if any) specified in the *TranscodingRequest* element. Having *applicationSizeLimit* at the Transcoding Request level avoids having to specify the parameter for each individual Transcoding Job within the request.

All the above size limit parameters are of type unboundedLong (see section 5.1.3). A value of -1 removes all limitations on the size (size is unbounded). A size limit value of 0 means that the relevant media type SHALL not appear in the output. All content element of this media type SHALL either be removed or transcoded into a different media type.

In case no size limit is specified in any level, the Transcoding Platform SHALL use the size limit from the capabilities profile, if given. If no capabilities profile is specified the default size limit value SHALL be -1 (unlimited).

5.1.8 Overall UML Diagram

The following UML diagram gives an overview of all the data structures used for the STI Transcoding Request and Response.

A fully detailed UML diagram can be found in Appendix C.

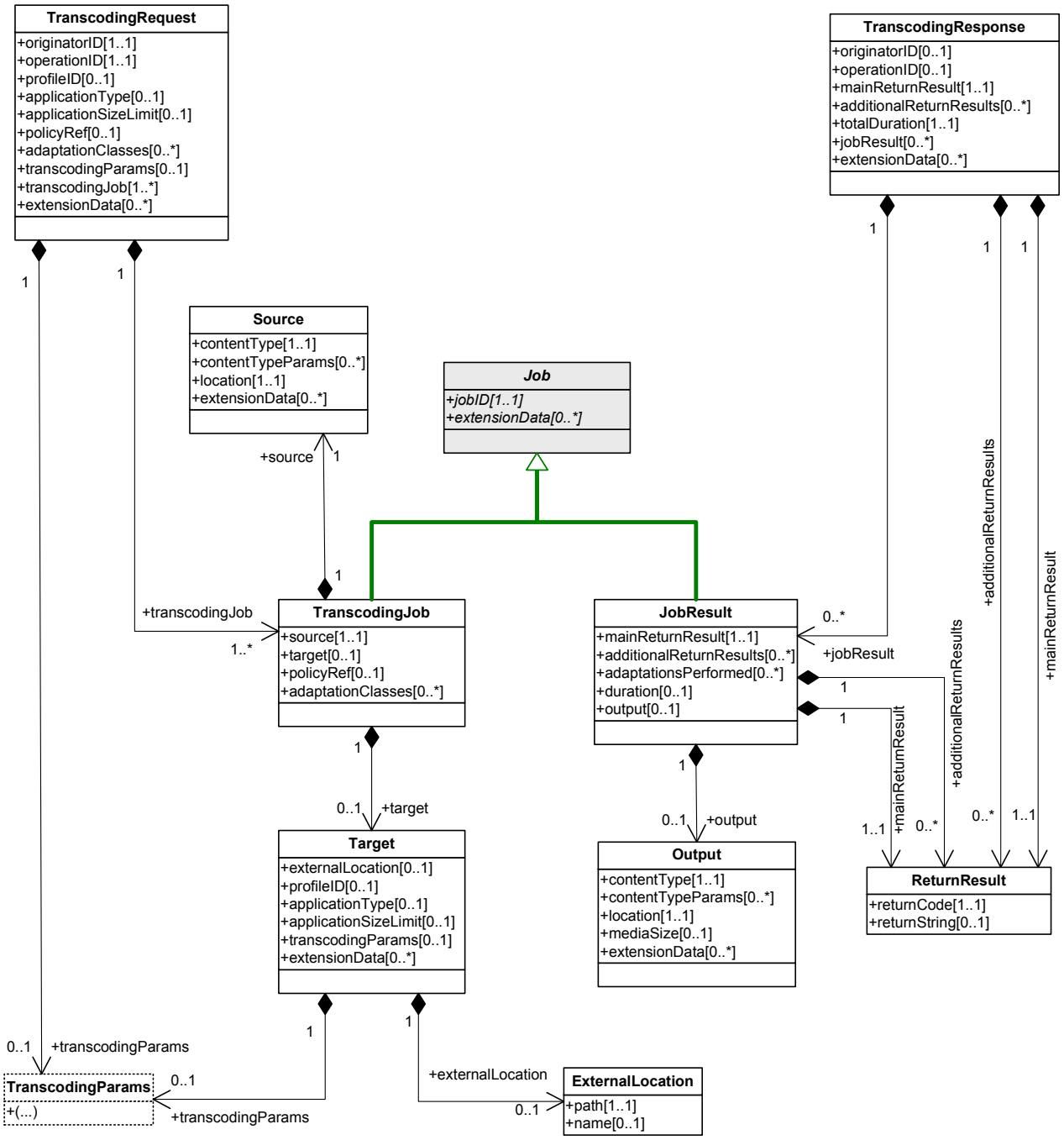

Figure 7Overall UML Diagram

5.2 Transcoding Request

5.2.1 High-level Format

As mentioned in sections 5.1.4 and 5.1.5, the Transcoding Requests SHALL contain a SOAP Envelope, made of one or more Transcoding Jobs and MAY contain one or more Content attachments (if some Content elements are contained in the request itself). All Content elements to be transcoded SHALL be referenced in the Transcoding Jobs (pointing to an attachment or an external source).

The Transcoding Request SHALL contain at least one Transcoding Job and MAY contain several Transcoding Jobs.

Figure 8 shows the structure of a Transcoding Request.

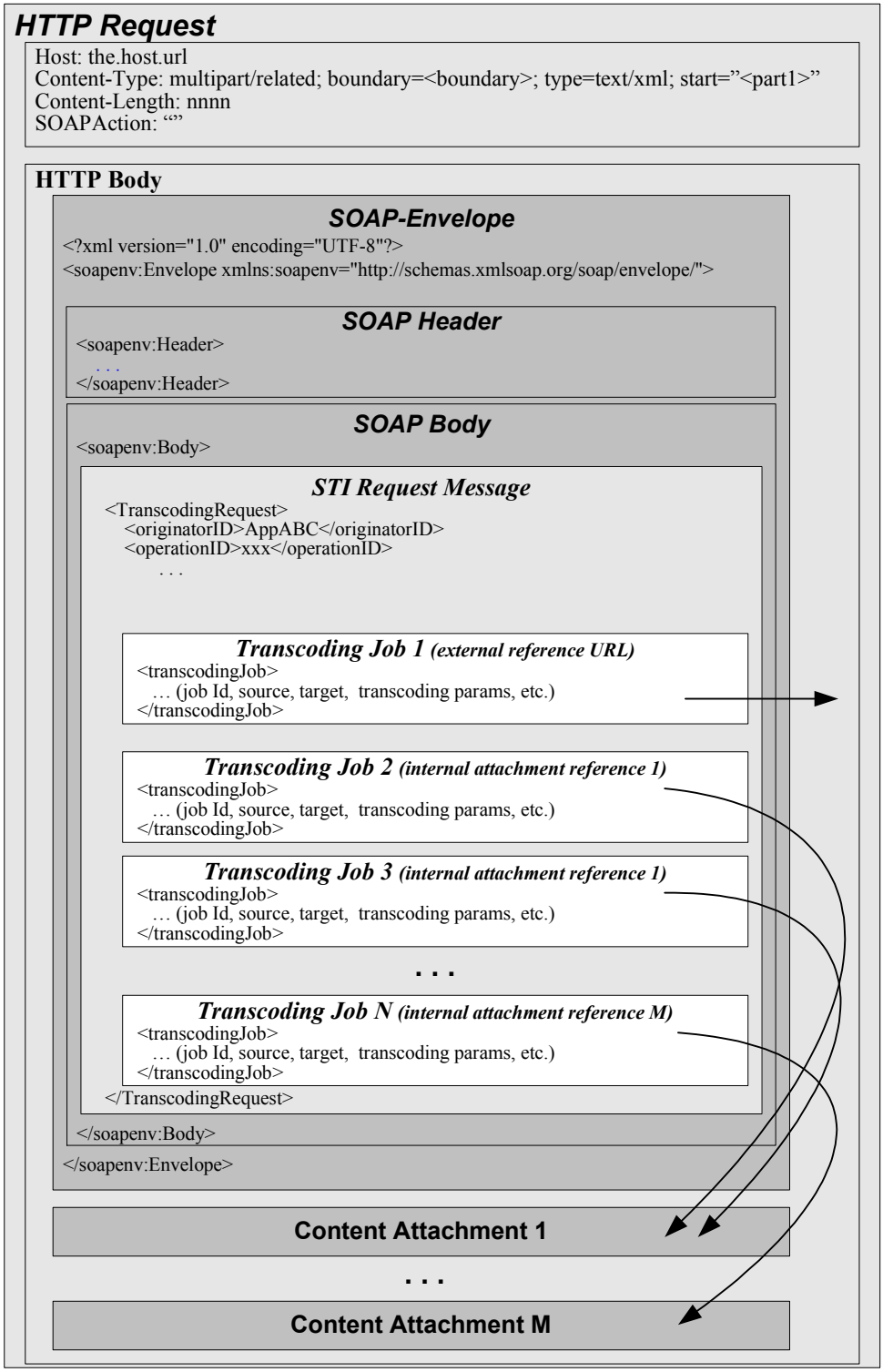

Figure 8STI Transcoding Request structure

The following sub-sections explain each one of the possible structures in more detail.

5.2.1.1 References to External Content Elements

Transcoding Requests that do not contain attachments (all the files are referenced using a URIs pointing to external storage) have HTTP headers and a single SOAP-based Request Body (contained in a SOAP-Envelope). These HTTP headers SHALL specify the target host, the content length (total content-length of the request), the content type and the SOAP action. It MAY include the content transfer encoding.

The following insert gives an example of the structure in that case:

```
POST /TranscodingRequest1 HTTP/1.1
Host: the.host.url
Content-Type: text/xml; charset=charset
Content-Length: nnnn
Content-Transfer-Encoding: 8bit
SOAPAction: “”

SOAP Envelope including the SOAP-based Request Body comes here (see section 5.2.2)
```

Notes:

a) A SOAP envelope MAY contain SOAP headers and SHALL contain a SOAP body.

b) The character set for the SOAP message SHALL be either UTF-8 or UTF-16. As a consequence of this, in conjunction with SOAP 1.1's requirement to use the text/xml content type (which has a default character encoding of "us-ascii"), the charset parameter SHALL always be present on the SOAP envelope's content type. A further consequence of this is that the encoding pseudo-attribute of XML declaration within the message SHALL always be ignored (i.e. the one in the following line, if present in the request: <?xml version="1.0" encoding="UTF-8" ?>)

c) The empty string "" SHOULD be used for SOAPAction, but implementations should allow for any value to be placed there so long as the receiving node does not process the contents beyond checking for its presence. The SOAPAction header is being discontinued in future versions of SOAP since it can misrepresent the actual content in the SOAP envelope. All current tools should be ignoring this header, and be using the information in the SOAP envelope instead.

5.2.1.2 References to Content Elements Contained in the Request

Transcoding Requests with Content attachments SHALL follow the definition of SOAP with Attachments (see [OMA MWS Guidelines], section 6.8).

Transcoding Requests with Content attachments SHALL contain the HTTP headers as defined in 5.2.1.1 and a string that will indicate the boundary between each Content attachment, as defined in MIME (see [MIME]). These HTTP headers SHALL specify the target host, the content-length (total content-length of the request), the content type and the SOAP action. They MAY include the transfer encoding.

Each Content attachment SHALL contain its own header (using [MIME]).

Each Content attachment header SHALL contain the unique ID (Content ID according to [RFC2392]) of the content, the content-type, and the content-transfer-encoding (e.g. base64, binary).

The first content attachment SHALL contain the SOAP-based Request Body (inside a SOAP-Envelope). All other content attachments SHALL either contain media element(s) to transcode, or additional data related to the request (e.g. Policy, proprietary extension data, etc.).

The following insert gives an example of the structure in that case:

```
POST /TranscodingRequest HTTP/1.1
Host: the.host.url
Content-Type: multipart/related; boundary=<the_boundary_string>; type=text/xml; start="<first-part>"
Content-Length: nnnn
SOAPAction: ""

--<the_boundary_string>
Content-Type: text/xml; charset="charset"
Content-ID: <first-part>
Content-Length: nnnn
Content-Transfer-Encoding: 8bit

  SOAP Envelope including the SOAP-based Request Body comes here (see section 5.2.2)

--<the_boundary_string>
Content-Type: image/png
Content-ID: <image1>
Content-Transfer-Encoding: binary

  Content Attachment Image comes here

--<the_boundary_string>--
```

See example on page 26 for notes about the usage of SOAP.

5.2.2 Request Body

5.2.2.1 Detailed Structure

The SOAP-based Request Body SHALL be structured as shown by the following table. This structure reflects the content of the XML Schema (.xsd) which can be found at [STI XSD]. In case of a conflict between the structures presented in this document (tables or UML) and the [STI XSD], implementations SHOULD follow the [STI XSD].

Text written in **bold** indicates a structure, which is defined further down in the table or in another table in this specification.

Table 1: Transcoding Request - Detailed Structure

No.	Parent	Name	Type	Possible Values	Mandatory/ Optional	Description
1.		**TranscodingRequest**			Mandatory (1..1)	Several Transcoding Jobs can be sent within one request.
2.	Transcoding Request	originatorID	Token		Mandatory (1..1)	A unique ID which represents the Application Platform that the request originated from. This parameter MAY be used for tracking, and it SHOULD NOT affect the Transcoding process.
3.	TranscodingRequest	operationID	Token		Mandatory (1..1)	A unique ID that is given by the Application Platform. IDs are not necessarily unique across all Application Platforms.
4.	TranscodingRequest	profileID	String		Optional (0..1)	The pre-defined Profile to be used for all the Transcoding

						Jobs within this request unless overwritten by *transcodingParams* or by *profileID* at the Transcoding Job level (see section 5.1.7.3). The User-Agent string, UAProf string (URL), or a proprietary string may be used. If the User-Agent or the UAProf of the target device are given in the request, the Transcoding Platform SHOULD recognize them. The *profileID* MAY be complemented by the *applicationType* parameter (see section 5.1.7.1).
5.	TranscodingRequest	applicationType	String		Optional (0..1)	The application for which the Transcoding is to be performed. This parameter MAY be used to complement the *profileID* parameter (see section 5.1.7.1). It can possibly lead to some application-specific Transcoding behavior. Possible values are, for example, "MMS" and "Browsing"
6.	TranscodingRequest	applicationSizeLimit	unboundedLong	In bytes	Optional (0..1)	See section 5.1.7.6 for details about the usage of the *sizeLimit* and *applicationSizeLimit* parameters.
7.	TranscodingRequest	policyRef	anyURI	URI	Optional (0..1)	The URI of a pre-defined Policy to be used for the request.
8.	TranscodingRequest	**adaptationClasses**			Optional (0..1)	Information about adaptation classes; which are of interest and if they are allowed or not.
9.	TranscodingRequest	**transcodingParams** (see ***Table 7***)			Optional (0..1)	The transcoding parameters to be used for all the Transcoding Jobs within this operation. When a predefined Profile is also specified, the values specified within the *transcodingParams* override the corresponding ones in the Profile (see section 5.1.7.3).
10.	TranscodingRequest	**transcodingJob**			Mandatory (1..n)	Structure representing a Transcoding Job. This element MAY appear multiple times.
11.	TranscodingRequest	**extensionData** (see ***Table 7***)			Optional (0..1)	Proprietary extension data.
12.	adaptationClasses	**adaptationClass**			Optional (0..n)	
13.	adaptationClass	className	Token		Mandatory (1..1)	Name of an adaptation class of interest to the application.

14.	adaptationClass	allowed	Boolean		Optional (0..1)	Specifies if, in a given job, a set of adaptations belonging to the given adaptation class is allowed or not. If the *allowed* parameter is not present, the adaptation class is allowed by default. For a given job, in case of contradiction between allowed and non-allowed adaptations, non-allowed adaptations take precedence. That is, an adaptation operation is forbidden if it would make the relevant Transcoding Job fall in an adaptation class which is forbidden (e.g. an action allowed in minor but forbidden in major would be forbidden).
15.	adaptationClass	classRef	anyURI		Optional (0..1)	The URI of a pre-defined adaptation class definition.
16.	transcodingJob	jobID	Token		Mandatory (1..1)	A unique ID that is given by the Application Platform to uniquely identify the Transcoding Job within a Transcoding Request. The globally unique identifier of a job is a combination of originatorID, operationID, and jobID
17.	transcodingJob	**extensionData** (see ***Table 7***)			Optional (0..1)	Proprietary extension data.
18.	transcodingJob	**source**			Mandatory (1..1)	The source for the Transcoding Job.
19.	transcodingJob	**target**			Optional (0..1)	The target of this Transcoding Job.
20.	transcodingJob	policyRef	anyURI	URI	Optional (0..1)	The URI of a pre-defined Policy to be used for the Transcoding Job. If a *policyRef* is specified both at the request and at the job level, the *policyRef* at the job level SHALL take precedence
21.	transcodingJob	**adaptationClasses**			Optional (0..1)	Information about adaptation classes; which are of interest and if they are allowed or not. If *adaptationClasses* is present at the Transcoding Job level, it SHALL take precedence over the one (if any) specified at the Transcoding Request level.
22.	source	contentType	Token	According to the content types	Mandatory (1..1)	The content type of the source.

				naming (see section **Error! Reference source not found.**)		
23.	source	**contentTypeParams** (see ***Table 7***)			Optional (0..1)	The content type parameters of the source (e.g. charset).
24.	source	location	anyURI	URI	Mandatory (1..1)	For referenced sources: The full-path (URL) to an external storage including the name of the resource. For sources included in the request: the Content ID (cid) of the attachment within the request [RFC2392].
25.	source	**extensionData** (see ***Table 7***)			Optional (0..1)	Proprietary extension data.
26.	target	**externalLocation**			Optional (0..1)	The path to an external storage. The Transcoding Platform SHALL return the full URI of the transcoded media in the *output* element of the *jobResult* (see ***Table 8***). If this field is not present, the transcoded content will be returned as an attachment included in the Transcoding Response (even if the source element is referenced).
27.	target	profileID	String		Optional (0..1)	A predefined Profile to use for this Transcoding Job. The *profileID* MAY be complemented by the *applicationType* parameter (see section 5.1.7.1).When a *profileID* is present at the request level, the *profileID* at the target level SHALL take precedence (see section 5.1.7.3).
28.	target	applicationType	String		Optional (0..1)	The application for which the Transcoding is to be performed. This parameter MAY be used to complement the *profileID* parameter (see section 5.1.7.1). It can possibly lead to some application-specific Transcoding behavior. When an *applicationType* is also present at the request level, the *applicationType* at the target level SHALL take precedence. Possible values are, for example, “MMS” and “Browsing”

29.	target	applicationSizeLimit	unboundedLong	In bytes	Optional (0..1)	See section 5.1.7.6 for details about the usage of the *sizeLimit* and *applicationSizeLimit* parameters.
30.	target	**transcodingParams** (see *Table 7*)			Optional (0..1)	Explicit transcoding parameters to be used for this Transcoding Job. See section 5.1.7.3 for details on *profileID* and *transcodingParams* hierarchy.
31.	target	**extensionData** (see ***Table 7***)			Optional (0..1)	Proprietary extension data.
32.	externalLocation	path	anyURI		Mandatory (1..1)	The path (URI) to an external storage where the target media is to be placed, excluding the name of the media.
33.	externalLocation	name	Token		Optional (0..1)	The name of the target media, without a file extension. The Transcoding Platform SHOULD add an appropriate file extension to the name of the output media, if such an extension exists. If this parameter is not present, the final name of the result media SHOULD be constructed from the specified path and the name of the source media, with an appropriate file extension.

5.2.3 Supported Media Types

The following table describes supported media types.

Table 2: Supported Media Types

No.	Media Type	Description
1.	Image	An image file. Explicit transcoding parameters SHALL be specified under the Image element.
2.	Video	A video file, which may also contain audio. Explicit transcoding parameters SHALL be specified under the Video element.
3.	Text	A text file. Explicit transcoding parameters SHALL be specified under the Text element.
4.	Audio	An audio file. Explicit transcoding parameters SHALL be specified under the Audio element.
5.	Multipart	A multipart file. Explicit transcoding parameters SHALL be specified under the Multipart element.

5.2.4 Content Types and Codecs

5.2.4.1 Content Types

Content types for which a registered MIME type exists SHOULD be represented using the registered MIME type (e.g. multipart/related, multipart/mixed, image/jpeg, image/gif, audio/amr, etc. See [IANA]).

The Transcoding Platform and Application Platform MAY also support other Content-Types that will then be defined between them.

The *contentType* field SHOULD NOT contain any parameters. The Transcoding Platform SHOULD ignore any parameters in the *contentType* field.

If the Transcoding Platform does not recognize the Content type it SHALL return an error.

5.2.4.2 Content Type Parameters

Content type parameters generally relate to file format parameters. Content type parameters SHALL be supplied using a Name-Value format. For content types that have registered parameters, the parameters' Name and Value SHOULD be used as specified (e.g., Name = 'charset', Value = 'ISO-8859-1' for content-type text/plain). The following table lists additional possible parameters and their possible values.

The Transcoding Platform and Application Platform MAY also support other content type parameters that will be defined between them.

Table 3: Content Type Parameters

No.	Content Type	Parameter Name	Possible Values	Value Constraints	Description
1.	video/3gpp, audio/3gpp, video/3gpp2, audio/3gpp2 and other similar formats	brand	3gp4,3gp5, 3gp6,3g2a, mp42, ...	Exact	The output brand
2.	application/smil	subset	SMIL-CONF-1_2, SMIL-3GPP-R4, SMIL-3GPP-R5, …	Exact	The SMIL subset to be used.
3.	image/jpeg	type	exif, jfif	Exact	The jpeg file format.
4.	video/3gpp, audio/3gpp, video/3gpp2, audio/3gpp2 video/mp4	composition-mode	progressive-download	Exact	The progressive-download value specifies that the output media is to be suitable for progressive download.
5.	video/3gpp, audio/3gpp, video/3gpp2, audio/3gpp2 and other similar formats	audio-boxtype	mp4a, sqcp, …	Exact	mp4a and sqcp are the possible values for the QCELP codec in 3GPP2 files as per the 3GPP2 specification, see [C.S0050-0 V1.0].

5.2.4.3 Codecs

Codecs for which a MIME type has been registered at [IANA] (e.g. a file format registration such as audio/amr, audio/amr-wb, or an RTP Payload registration such as video/h263, audio/g723, etc.) SHOULD be represented using that MIME type. The following table lists some additional widely used codecs and their representation in STI.

The Transcoding Platform and Application Platform MAY also support other codecs that will then be defined between them.

If the Transcoding Platform does not recognize the codec it SHALL return an error.

Table 4: Codecs

No.	Codec Description	Codec string	Comment
1.	MPEG1 audio	audio/mpeg1	See [RFC3003]
2.	MPEG2 audio	audio/mpeg2	See [RFC3003]
3.	g72x audio codecs	audio/g72x	Some of these can be found at [IANA]
4.	PCM	audio/pcm	
5.	MPEG4-AAC	audio/aac	See [ISO/IEC 14496-3]
6.	MPEG2-AAC	audio/mpeg2-aac	
7.	WMA Codecs	audio/wma7	
8.		audio/wma8	
9.	MPEG4 video	video/mpg4	See [ISO/IEC 14496-2]
10.	H.264 / AVC	video/h264	See [ISO/IEC 14496-10]
11.	AVI codecs	video/<fourcc>	As listed in [FOURCC]
12.	WMV Codecs	video/wmv7	
13.		video/wmv8	
14.		video/wmv9	
15.			

5.2.4.4 Codec Parameters

Codec parameters SHALL be supplied using a Name-Value format. For MIME types that have registered parameters, the parameters' Names and Values SHOULD be used as specified (e.g. Name = profile, Value = 3 for codec video/h263-2000). See [IANA].

The following table lists additional codec parameters and their possible values.

The Transcoding Platform and Application Platform MAY also support other codec parameters that will then be defined between them.

Table 5: Codec Parameters

No.	Codec	Parameter Name	Possible Values	Value Constraints	Description
1.	audio/*	encoding-method	FBR, CBR, VBR	Exact	FBR = Fixed Bit Rate, CBR = Constant Bit Rate, VBR = Variable Bit Rate. FBR and CBR are the same
2.	audio/*	min-bit-rate	Non negative integer (in bits per second)	Minimum + Desired	The lower limit of a range of bit rates when Variable Bit Rate (VBR) is used. See Appendix I for a description of the relationship between min-bit-rate, bitRate, and encoding-method.
3.	audio/mpeg1	layer	3	Exact	
4.	audio/mpeg2	layer	2.5, 3	Exact	
5.	audio/aac	object-type	lc, ltp, ssr, scalable, he-aac, etc.	Exact	See [ISO/IEC 14496-3]
6.	image/jpeg	mode	sequential, progressive, lossless, hierarchical	Exact	
7.	image/jpeg, image/png	quality-factor	Integer between 0 and 100	Exact	See [ISO/IEC 10918-1]
8.	video/*	key-frame-spacing-ms	Non negative integer (in milliseconds)	Maximum + Desired	Maximum time interval between key frames (in milliseconds)
9.	video/*	key-frame-spacing-frames	Non negative integer	Maximum + Desired	Maximum number of frames between key frames.

10.	video/mpg4	profile	simple, simple-scalable, etc.	Exact	See [ISO/IEC 14496-2]
11.	video/mpg4	level	0, 1, etc.	Exact	See [ISO/IEC 14496-2]

5.2.5 Transformations

In order to perform transformations on the media the Application Platform SHALL specify, for each requested transformation, the type of the transformation, and the attributes for the transformation.

The following table lists the set of standard transformation algorithms, each with a non-extensible list of attributes, and the name (or names) of media the elements within the Transcoding Request in which these transformations can appear. The extensionData element MAY be used for proprietary transformations.
Note that transformations MAY also be used along with *transcodingParams* in order to obtain specific output. For more details, refer to Appendix H.

Support of the transformation algorithms in the following table is optional.

When a transformation (which is supported by the Transcoding Platform) fails to be executed – the Job MAY either succeed with an appropriate warning in the *jobResult* element, or fail with an error code. This behavior need not be consistent for all transformation algorithms.
Each of the specified attributes' values may have one of the following constraints:

Desired = The desired value.

Exact = The exact value that must be met.

Maximum = The value can't be exceeded.

Minimum = The value must be met or exceeded.

Desired+Maximum = The desired value, which cannot be exceeded.

Table 6: Transformations

No.	Transformation Type	Target Media Elements	Attribute Name	Possible attribute values	Mandatory / Optional attribute	Value Constraints	Description
1.	LevelCorrection	Image, VideoVisual	No attribute				Perform level correction.
2.	Mirror	Image, VideoVisual	axis	"UD", "LR"	M	Exact	"UD" (Up-Down) - vertical mirror "LR" (Left-Right) - horizontal mirror.
3.	NoiseReduction	Image, VideoVisual	No attribute				Perform noise reduction.
4.	Rotation	Image, VideoVisual	clockwiseAngle	90 180 270 auto	M	Exact (Except for the 'auto' value)	Perform Image rotation. "auto" rotation allows rotation based on aspect ratio. Automatic rotation allows the Transcoding Platform to perform a 90 degrees clockwise rotation, in order to maintain aspect ratio as close as possible to the original image (e.g. Transcoding image of 176x144 to 144x176 with "auto" clockwiseAngle will invoke the rotation). The "auto" rotation is meaningful only when

							requested aspect ratio is well defined (in TranscodingParams, profile or policy)[1].
5.	Sharpen	Image, VideoVisual	No attribute				Perform image sharpening.
6.	DurationLimit	Video, Audio, Multipart	limit	a positive integer value	M	Maximum	Truncate the media if its duration exceeds <limit> milliseconds.
7.	AGC	Audio, VideoAudio	No attribute				Perform Automatic Gain Control
8.	Offset	Image, Audio, Video	startTime	a positive integer value	M		The offset from the beginning of the source media to start the Transcoding from, in milliseconds.
9.	FrameRateSample	Image, Multipart[2]	fps	a positive float value	M	Desired	Change the frame sampling rate (without changing the speed and the duration of the media). Changes the number of frames/slides.
10.	FrameRateOutput	Image, Multipart[2]	fps	a positive float value	M	Desired	Change the frame rate in the output (and thus change also the speed and the duration of the media, e.g. to create slow/fast motion effects/adjustments). Does not change the number of frames/slides. For animation, this determines the time interval for each frame.
11.	NumberOfFrames	Image, Multipart[2]	totalFrames	a positive integer value	M	Exact	Change the number of frames/slides in the output. For Image, indicates whether animation (>1) or single image (=1) is intended.
12.	Cropping	Image, VideoVisual					Perform image cropping. A parameter list defines a rectangular region in source image coordinates[1].
			top	a non negative integer value	M	Exact	Top Y coordinate of the rectangle.
			left	a non negative integer value	M	Exact	Left X coordinate of the rectangle.
			bottom	a non negative integer value	M	Exact	Bottom Y coordinate of the rectangle.

[1] Unless specifically defined otherwise, the Rotation and Cropping transformations SHOULD be applied prior to the actual transcoding.

[2] In order to specify frame rate for video, use the *frameRate* parameter in the *VideoVisual* element.

			right	a non negative integer value	M	Exact	Right X coordinate of the rectangle.
13	FrameFill	Image, VideoVisual					Indicates if the target image frame should be filled while source image aspect ratio will be maintained, overriding either target height or width (when applicable). The parameters, if exists, define the chosen color, using RGB values.
			R	Integer in range 0..255	M	Exact	Value of red component
			G	Integer in range 0..255	M	Exact	Value of green component
			B	Integer in range 0..255	M	Exact	Value of blue component
14	Brightness	Image, VideoVisual	level	Integer in the range -50..50[3] (inclusive)	M	Desired	Brightness correction for output image. 0 indicates no correction.
15	Contrast	Image, VideoVisual	level	Integer in the range -50..50[3] (inclusive)	M	Desired	Contrast correction for output image. 0 indicates no correction.
16	Color	Image, VideoVisual	level	Integer in the range -50..50[3] (inclusive)	M	Desired	Color correction for output image. 0 indicates no correction.

5.2.6 TranscodingParams structure

5.2.6.1 UML Diagram

The following UML diagram gives an overview of the data structure used for the *transcodingParams* element of the request.

A fully detailed UML diagram can be found in Appendix C.

[3] The scale (-50..50) is a relative scale of intensity or level of the correction, and it is not defined by specific physical measure.

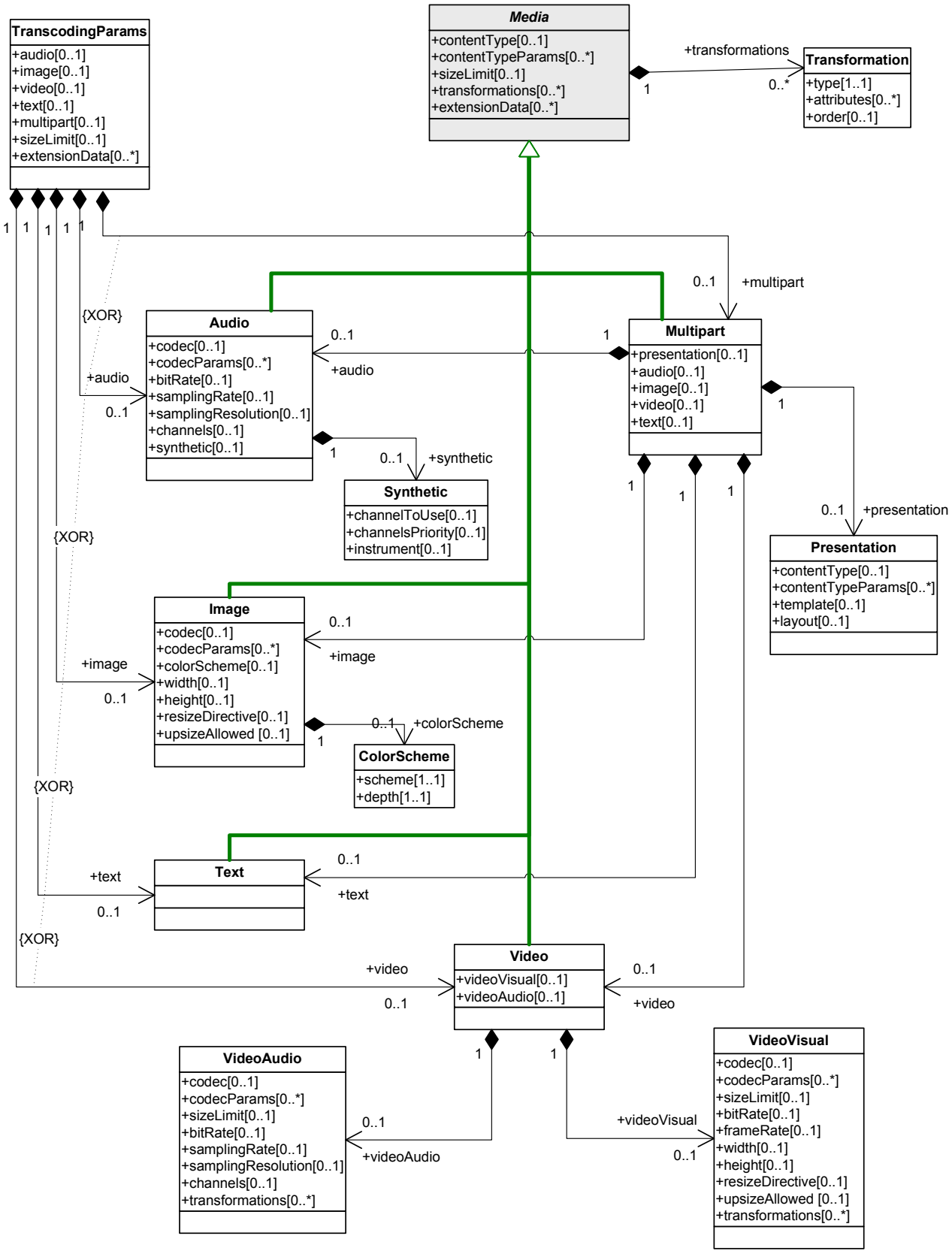

Figure 9 Transcoding Params UML Diagram

5.2.6.2 Detailed Structure

The following table contains the *transcodingParams* structure that can be specified in a Transcoding Request.

The table defines a number of structures separated by grey lines. Text written in **bold** indicates a structure, which is defined further down in the table or in another table in this specification.

For the definition of the constraints in the value constraints column see section 5.2.5.

Note that the *transcodingParams* MAY be complemented with some transformations to obtain specific output. See examples in Appendix H.

Table 7: Transcoding Parameters - Detailed Structure

No.	Parent	Name	Type	Units/ Possible Values	Mandatory/ Optional	Value Constraints	Description
1.		**transcodingParams**			Optional (0..1)		The transcoding parameters. When a predefined Profile is specified and this tag appears, the values specified within this tag override the ones in the Profile. The *transcodingParams* element SHALL contain at most one of the elements *audio / video / image / text / multipart*.
2.	transcodingParams	**audio**			Optional (0..1) (These elements are mutually exclusive)		The audio transcoding parameters.
3.	transcodingParams	**image**					The image transcoding parameters (also graphics).
4.	transcodingParams	**video**					The video transcoding parameters.
5.	transcodingParams	**text**					The text transcoding parameters.
6.	transcodingParams	**multipart**					The multipart transcoding parameters.
7.	transcodingParams	sizeLimit	unboundedLong	In bytes	Optional (0..1)	Maximum	The limit of the file sizes for the Transcoding Request or the Transcoding Job. See section 5.1.7.6 for details about the usage of the *sizeLimit* parameters.
8.	transcodingParams	**extensionData**			Optional (0..1)		Proprietary extension data.
9.	extensionData	**property**			Optional (0..n)		Proprietary extension parameters. All extension parameters SHALL be defined between the Application Platform and the Transcoding Platform. If the Transcoding Platform does not recognize an *extensionData* parameter's value it SHALL ignore the parameter and

							include a warning in the response. Note that these parameters can also include references to external data structure or an attachment.
10.	audio	contentType	Token	According to the content types naming (see section **Error! Reference source not found.**)	Optional (0..1)	Exact	The requested content type for the target (see section 5.2.4.1).
11.	audio	**contentTypeParams**			Optional (0..1)	Exact	Specific content type parameters (see section 5.2.4.2)
12.	audio	sizeLimit	unboundedLong	In bytes	Optional (0..1)	Maximum	Size limit for the audio. See section 5.1.7.6 for details about the usage of the *sizeLimit* parameters.
13.	audio	**transformations**			Optional (0..1)		The transformations that the content should go through.
14.	audio	**extensionData**			Optional (0..1)		Proprietary extension data.
15.	audio	codec	Token		Optional (0..1)	Exact	See section 5.2.4.3
16.	audio	**codecParams**			Optional (0..1)	Exact	See section 5.2.4.4
17.	audio	bitRate	nonNegativeInt	In bps	Optional (0..1)	Desired+Maximum	The desired bit-rate. Restricted by the codec[4].
18.	audio	samplingRate	nonNegativeInt	In Hz	Optional (0..1)	Desired+Maximum	The desired sampling rate. Restricted by the codec
19.	audio	samplingResolution	nonNegativeInt	In bits per sample	Optional (0..1)	Desired+Maximum	The desired sampling resolution. Restricted by the codec
20.	audio	channels	Token	"Mono", "Stereo", "DualMono", "IntensityStereo"	Optional (0..1)	Exact	

[4] See Appendix I for a description of the relationship between the bitRate parameter and the min-bit-rate and encoding-method codec parameters.

[5] This list is not exhaustive and other values can be used in order to express additional and/or proprietary information.

				[5]			
21.	audio	**synthetic**			Optional (0..1)		Synthetic audio related parameters.
22.	synthetic	channelToUse	nonNegativeInt		Optional (0..1)	Exact	Number of the input channel to be used in case the output contains only one channel.
23.	synthetic	channelsPriority	Token		Optional (0..1)	Exact	A comma separated permutation of the channels, representing the priority of the channels, from the most important to the least important. White spaces are not allowed.
24.	synthetic	instrument	nonNegativeInt		Optional (0..1)	Exact	The instrument to use, when transcoding to midi. Irrelevant for other formats.
25.	contentTypeParams	**property**			Optional (0..n)		For the name and value of the content type parameters (see section 5.2.4.2).
26.	codecParams	**property**			Optional (0..n)		For the name and value of the codec parameters (see section 5.2.4.4).
27.	transformations	**transformation**			Optional (0..n)		For a list of standard transformations see section 5.2.5.
28.	transformation	type	Token	According to the Transformations naming (see section 5.2.5)	Mandatory (1..1)	Exact	Type of transformation
29.	transformation	**attributes**			Optional (0..1)		The attributes of the transformation. Standard transformations SHALL support only those attributes that are listed in section 5.2.5.
30.	transformation	order	nonNegativeInt		Optional (0..1)		In the case of multiple transformations of the same media object – the transformations that do include an *order* value SHOULD be performed by order of this value (starting with lower *order* values). The order of transformations with the same order value is undefined. So is the order of transformations that do not

							specify their order. In these cases the Transcoding Platform MAY choose the order as it sees best.
31.	attributes	**property**			Optional (0..n)		For a list of supported attributes and their corresponding values for standard transformations see section 5.2.5.
32.	property	name	Token		Mandatory (1..1)		
33.	property	value	String		Optional (0..1)		
34.	image	contentType	Token	According to the content types naming (see section **Error! Reference source not found.**)	Optional (0..1)	Exact	The requested content type for the target (see section 5.2.4.1).
35.	image	**contentTypeParams**			Optional (0..1)	Exact	Specific content type parameters (e.g. Codec). See section 5.2.4.2.
36.	image	sizeLimit	unboundedLong	In bytes	Optional (0..1)	Maximum	Size limit for the image. See section 5.1.7.6 for details about the usage of the *sizeLimit* parameters.
37.	image	**transformations**			Optional (0..1)		The transformations that the content should go through.
38.	image	**extensionData**			Optional (0..1)		Proprietary extension data.
39.	image	codec	Token		Optional (0..1)	Exact	See section 5.2.4.3
40.	image	**codecParams**			Optional (0..1)	Exact	See section 5.2.4.4
41.	image	**colorScheme**			Optional (0..1)	Exact	The desired color scheme
42.	image	width	nonNegativeInt	In pixels	Optional (0..1)	Desired+Maximum	The target width. The default is the original width.
43.	image	height	nonNegativeInt	In pixels	Optional (0..1)	Desired+Maximum	The target height. The default is the original height.
44.	image	resizeDirective	Token	"AspectRatio",	Optional (0..1)	Exact.	**AspectRatio** - Maintain the aspect ratio when resizing.

				"Crop", "Stretch"[6]			This might override one of the image's *width*/*height* parameters. Note that this directive can be complemented with the FrameFill transformation (see ***Table 6***) **Crop** – Applicable when shrinking. Take the image's central rectangle. **Stretch** – Fit the entire image into the new dimensions. The default is AspectRatio.
45.	image	upsizeAllowed	Boolean		Optional (0..1)	Exact	For cases where the source media is smaller (in resolution) than the target, this parameter specifies whether upsizing is allowed or not. If true, the *resizeDirective* SHALL be considered for the upsizing (AspectRatio or Stretch). If false, the target width and height parameters SHALL be ignored. Default is false.
46.	colorScheme	scheme	Token	"True", "Palette Color" "Palette Grey"[7]	Mandatory (1..1)	Exact	
47.	colorScheme	depth	nonNegativeInt		Mandatory (1..1)	Exact	If Scheme is "True": number of bits per pixel If Scheme is "Palette…": maximum number of colors in palette
48.	video	contentType	Token	According to the content types naming (see section **Error! Reference source not found.**)	Optional (0..1)	Exact	The requested content type for the target (see section 5.2.4.1).

[6] This list is not exhaustive and other values can be used in order to express additional and/or proprietary information.

[7] This list is not exhaustive and other values can be used in order to express additional and/or proprietary information.

49.	video	**contentTypeParams**			Optional (0..1)	Exact	Specific content type parameters (e.g. Codec). See section 5.2.4.2.
50.	video	sizeLimit	unboundedLong	In bytes	Optional (0..1)	Maximum	Size limit for the video. See section 5.1.7.6 for details about the usage of the *sizeLimit* parameters.
51.	video	**transformations**			Optional (0..1)		The transformations that the content should go through.
52.	video	**extensionData**			Optional (0..1)		Proprietary extension data.
53.	video	**videoVisual**			Optional (0..1)		Video visual parameters.
54.	video	**videoAudio**			Optional (0..1)		Video Audio parameters.
55.	videoVisual	codec	Token		Optional (0..1)	Exact	See section 5.2.4.3
56.	videoVisual	**codecParams**			Optional (0..1)	Exact	See section 5.2.4.4
57.	videoVisual	sizeLimit	unboundedLong	In bytes	Optional (0..1)	Maximum	Size limit for the video's visual part. Value 0 means remove the visual part. See section 5.1.7.6 for details about the usage of the *sizeLimit* parameters.
58.	videoVisual	width	nonNegativeInt	In pixels	Optional (0..1)	Desired+Maximum	Target maximum video width
59.	videoVisual	height	nonNegativeInt	In pixels	Optional (0..1)	Desired+Maximum	Target video height
60.	videoVisual	frameRate	Float	In frames per second	Optional (0..1)	Maximum	The maximum frame rate. Sub 1 fps and non-rounded values are enabled.
61.	videoVisual	bitRate	nonNegativeInt	In bps	Optional (0..1)	Maximum	The target maximum bit rate.
62.	videoVisual	resizeDirective	Token	"AspectRatio", "Crop", "Stretch"[8]	Optional (0..1)	Exact.	**AspectRatio** - Maintain the aspect ratio when resizing. This might override one of the image's *width*/*height* parameters. Note that this directive can be complemented with the FrameFill transformation (see ***Table 6***) **Crop** – Applicable when shrinking. Take the image's central rectangle. **Stretch** – Fit the entire image into the new

[8] This list is not exhaustive and other values can be used in order to express additional and/or proprietary information.

							dimensions. The default is AspectRatio.
63.	videoVisual	upsizeAllowed	Boolean		Optional (0..1)	Exact	For cases where the source media is smaller (in resolution) than the target, this parameter specifies whether upsizing is allowed or not. If true, the *resizeDirective* SHALL be considered for the upsizing (AspectRatio or Stretch). If false, the target width and height parameters SHALL be ignored. Default is false.
64.	videoVisual	**transformations**			Optional (0..1)		The transformations that the content should go through.
65.	videoAudio	codec	Token		Optional (0..1)	Exact	See section 5.2.4.3
66.	videoAudio	**codecParams**			Optional (0..1)	Exact	See section 5.2.4.4
67.	videoAudio	sizeLimit	unboundedLong	In bytes	Optional (0..1)	Maximum	Size limit for the video's audible part. See section 5.1.7.6 for details about the usage of the *sizeLimit* parameters. Value 0 means remove the audible part.
68.	videoAudio	bitRate	nonNegativeInt	In bps	Optional (0..1)	Desired+Maximum	The maximum bit-rate. Restricted by the codec[9]
69.	videoAudio	samplingRate	nonNegativeInt	In Hz	Optional (0..1)	Desired+Maximum	The desired sampling rate. Restricted by the codec
70.	videoAudio	samplingResolution	nonNegativeInt	In bits per sample	Optional (0..1)	Desired+Maximum	The desired sampling resolution. Restricted by the codec
71.	videoAudio	channels	Token	"Mono", "Stereo", "DualMono", "IntensityStereo"[10]	Optional (0..1)	Exact	
72.	videoAudio	**transformations**			Optional (0..1)		The transformations that the content should go through.
73.	text	contentType	Token	According to the	Optional (0..1)	Exact	The requested content type for the target (see section

[9] See Appendix I for a description of the relationship between the bitRate parameter and the min-bit-rate and encoding-method codec parameters.

[10] This list is not exhaustive and other values can be used in order to express additional and/or proprietary information.

				content types naming (see section **Error! Reference source not found.**)			5.2.4.1).
74.	text	**contentTypeParams**			Optional (0..1)	Exact	Specific content type parameters (see section 5.2.4.2).
75.	text	sizeLimit	unboundLong	In bytes	Optional (0..1)	Maximum	Size limit for the text part. See section 5.1.7.6 for details about the usage of the *sizeLimit* parameters.
76.	text	**transformations**			Optional (0..1)		The transformations that the content should go through.
77.	text	**extensionData**			Optional (0..1)		Proprietary extension data.
78.	multipart	contentType	Token	According to the content types naming (see section **Error! Reference source not found.**)	Optional (0..1)	Exact	The requested content type for the target (see section 5.2.4.1).
79.	multipart	**contentTypeParams**			Optional (0..1)	Exact	Specific content type parameters (see section 5.2.4.2).
80.	multipart	sizeLimit	unboundedLong	In bytes	Optional (0..1)	Maximum	Size limit for the whole multipart. See section 5.1.7.6 for details about the usage of the *sizeLimit* parameters.
81.	multipart	**transformations**			Optional (0..1)		The transformations that the content should go through.
82.	multipart	**extensionData**			Optional (0..1)		Proprietary extension data.
83.	multipart	**presentation**			Optional (0..1)		Parameters for the target presentation.
84.	multipart	**audio**			Optional (0..1)		Audio transcoding parameters for all audio parts of the multipart.
85.	multipart	**image**			Optional (0..1)		Image transcoding parameters for all image parts of the multipart.
86.	multipart	**video**			Optional		Video transcoding

					(0..1)		parameters for all video parts of the multipart.
87.	multipart	**text**			Optional (0..1)		Text transcoding parameters for all text parts of the multipart.
88.	presentation	contentType	Token	According to the content types naming (see section **Error! Reference source not found.**), or "none"	Optional (0..1)	Exact	The requested content type for the target (see section 5.2.4.1). "none" will remove the presentation.
89.	presentation	**contentTypeParams**			Optional (0..1)	Exact	Specific content type parameters (see section 5.2.4.2).
90.	presentation	template	anyURI		Optional (0..1)		Template for presentation to be used in case of presentation format change. Used as a complement to the existing presentation or may replace it based on policies. This is either a URI pointing to an external storage, or a content ID of the attachment
91.	presentation	layout	Token	"Portrait", "Landscape"	Optional (0..1)	Exact	Target layout. Precedence: 1) Presentation Layout, 2) Layout inside Presentation Template 3) Layout of the source input presentation file. 4) Default: Portrait.

Note:

Since a *profileID* can be specified, all the explicit transcoding parameters related tags are optional. If a *profileID* is not specified, then all the transcoding parameters relevant to the Transcoding Job SHOULD be specified.

5.3 Transcoding Response

The result of a transcoding operation is returned to the request sender using a *TranscodingResponse* structure embedded in the body of a SOAP response message. The same *TranscodingResponse* structure is used for a successful operation result and for a failure operation result.

An STI operation SHALL be considered successful unless there were one or many errors during the handling of the operation that prevented the completion of the whole operation. Note that the triggering of a failure can be different from one Transcoding Platform to another.

The Transcoding Response SHALL be successful even if some (but not all) of its Transcoding Jobs failed. A failure of **all** the Transcoding Jobs within a Transcoding Request SHALL cause the whole operation to fail (as detailed in section 5.3.2).

5.3.1 Successful Transcoding Response – High-level overview

The structure of a successful Transcoding Response is quite similar to the Transcoding Request – HTTP headers, a SOAP-Envelope containing one or more Job Results and optionally the Content attachments. The *TranscodingResponse* structure is located inside the SOAP message body part.

Figure 10 shows an example of the structure for a successful Transcoding Response.

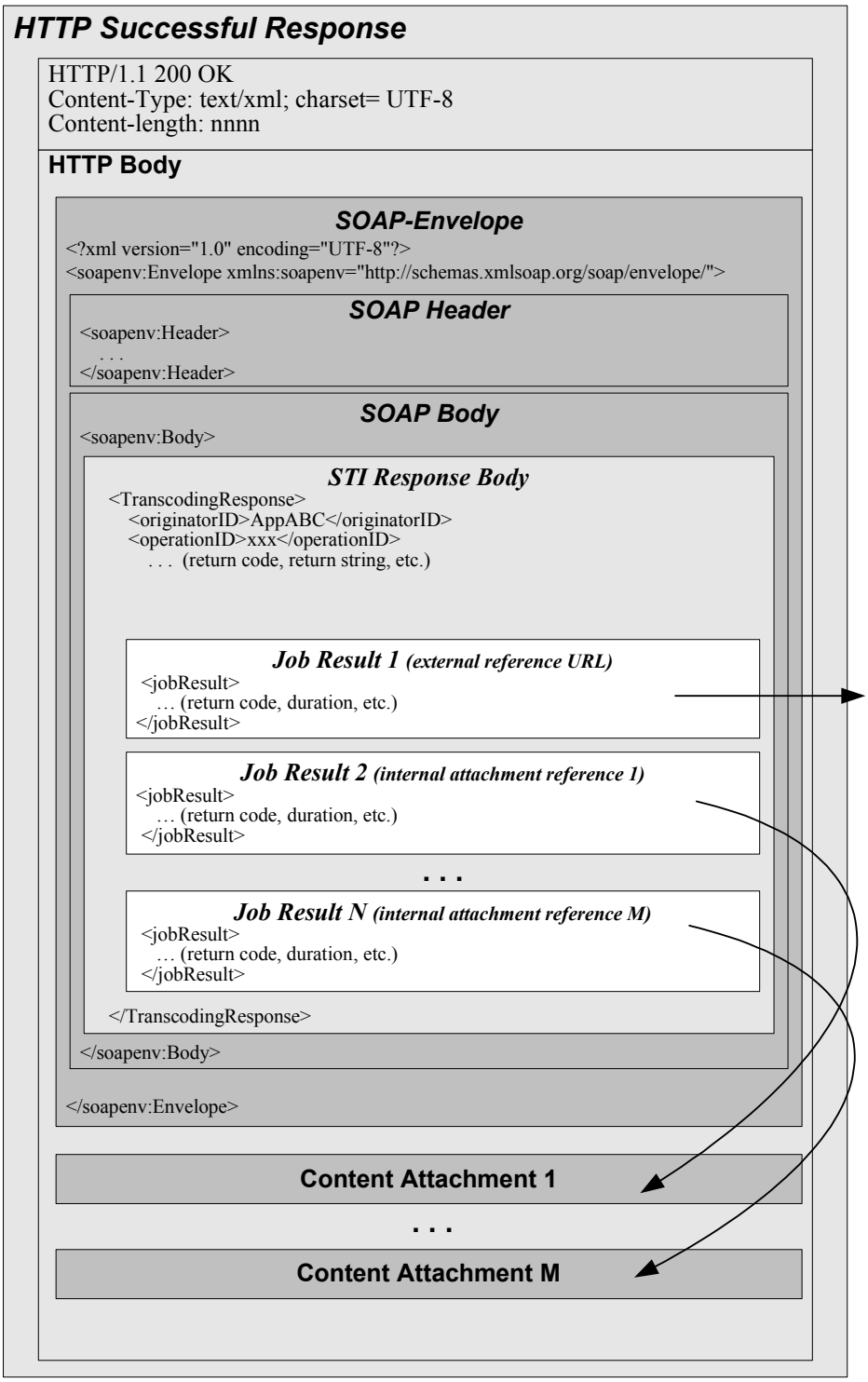

Figure 10 STI Transcoding Response Structure

5.3.1.1 HTTP Response Information Associated with Successful Transcoding Responses

Since STI is a basic web service using SOAP messages over HTTP protocol to send requests and return responses asynchronously, the response mechanism at the HTTP level SHALL be compliant with the requirements described in the Web Services Interoperability Organization documents:

- "Basic Profile Version 1.0" as referred in [OMA MWS Guidelines], section 4.
- "Attachments Profile 1.0" [WS-I Attachment Profile 1.0], section 3.
- "Simple SOAP Binding Profile Version 1.0" [WS-I SOAP Binding Profile 1.0], section 3

All these documents as defined by the WebServices-Interoperability organization [WS-I]. The use of these documents is a recommendation of the OMA-MWS group.

The HTTP headers and header values that SHOULD be or MUST be returned are defined in the Web Services Interoperability Organization documents listed above. The HTTP headers SHALL specify the content type, the content length (total content-length of the response),

Most of the time, the HTTP status code returned with a successful SOAP response message is "200 OK". However, the web infrastructure MAY respond with other informational codes (1xx) for intermediate results, successful/warning codes (2xx), or redirection codes (3xx) according to the scenarios described in the Web Services Interoperability Organization documents listed above. This is beyond the control of the Transcoding Platform. Note that codes 1xx and 3xx would probably not contain any SOAP message. These codes would be handled by the web infrastructure, not by the Transcoding Platform, and are out of the scope of STI 1.0.

Following is an example of an HTTP/SOAP successful response message that does not contain any attachment:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8
Content-length:  nnnn
. . .

SOAP Envelope including the SOAP-based Successful Response Body comes here (see sections 5.3.1.2 and 5.3.3)
```

Following is an example of an HTTP/SOAP successful response message that contains some attachments:

```
HTTP/1.1 201 OK
Content-Type: multipart/related; boundary=<the_boundary_string>; type=text/xml; start="<first-part-id>"
Content-length:  nnnn
. . .

--<the_boundary_string>
Content-Type: text/xml; charset=UTF-8
Content-ID: <first-part-id>
Content-Length: nnnn
Content-Transfer-Encoding: 8bit
```

SOAP Envelope including the SOAP-based Successful Response Body comes here (see sections 5.3.1.2 and 5.3.3)

--<the_boundary_string>

Content-Type: image/jpg

Content-ID: <content-name>

Content-Length: nnnn

Content-Transfer-Encoding: binary

{Content attachment data comes here}

--<the_boundary_string>--

Note:

The character set for the SOAP envelope SHALL be either UTF-8 or UTF-16. As a consequence of this, in conjunction with SOAP 1.1's requirement to use the text/xml content type (which has a default character encoding of "us-ascii"), the charset parameter SHALL always be present on the SOAP envelope's content type. A further consequence of this is that the encoding pseudo-attribute of XML declaration within the message SHALL always be ignored (i.e. the one in the following line, if present in the response: <?xml version="1.0" encoding="UTF-8" ?>)

5.3.1.2 SOAP Message Information Associated with Successful Transcoding Responses

The XML representation of a successful SOAP response message has constraints similar to a request message as defined in the Web Services Interoperability Organization documents listed in section 5.3.1.1.

The successful response message MAY contain references to either external content elements or self-contained content elements. The same differences between a Transcoding Request with contained content elements, and without contained content elements, as discussed for the Transcoding Request (section 5.2.1), apply for a successful Transcoding Response.

Figure 10 shows an example of the structure for a successful Transcoding Response. The SOAP-Envelope is returned as part of the HTTP response main body when no attachments are provided with the response or as part of the HTTP response first body part when some attachments are provided with the response.

The data specific to a successful response SHALL be contained directly under the SOAP Body element of the SOAP envelope.

5.3.2 Failed Transcoding Response - High-level Overview

A failed Transcoding Response is returned in a situation where the whole Transcoding Request could not be handled. This can happen if there was a problem at the HTTP or the SOAP level, a problem with the Transcoding Request parameters, or any other problem, which relates to the whole operation (and not only to some of the Transcoding Jobs within the Transcoding Request). A failure of **all** the Transcoding Jobs within a Transcoding Request SHALL also result in a failure response.

The following contains additional information for the errors at the various levels:

- Errors at the HTTP level
  HTTP responses with 5xx status codes SHOULD contain a body unless one of the following cases occurs:
    - A problem at the HTTP level

- An unidentifiable problem at the SOAP level
- An IO error while generating the response

- Errors at the SOAP level

  Failed Transcoding Responses caused by an error at the SOAP level SHOULD contain a SOAP message. That SOAP message SHOULD contain a SOAP Fault element in the SOAP body as shown in section 5.3.2.2.

- Errors at the Transcoding Request level

Failed Transcoding Responses (with a 4xxx or 5xxx STI error code) SHALL be returned in an HTTP response with a 500 status code. These HTTP responses SHALL contain a SOAP message with the SOAP Fault element in the SOAP body. The structure of a failed Transcoding Response is similar to a Transcoding Request and to a successful Transcoding Response. It contains HTTP headers, SOAP-Envelope and a SOAP Body. However, the SOAP Body contains the SOAP Fault element, and the STI *TranscodingResponse* structure is located inside the <detail> element of the SOAP Fault body part.

For more details about the SOAP Fault element and its use in the context of STI, refer to sections 5.3.2.2 and 5.3.2.3 below.

Figure 11 shows an example of the structure for a failed Transcoding Response.

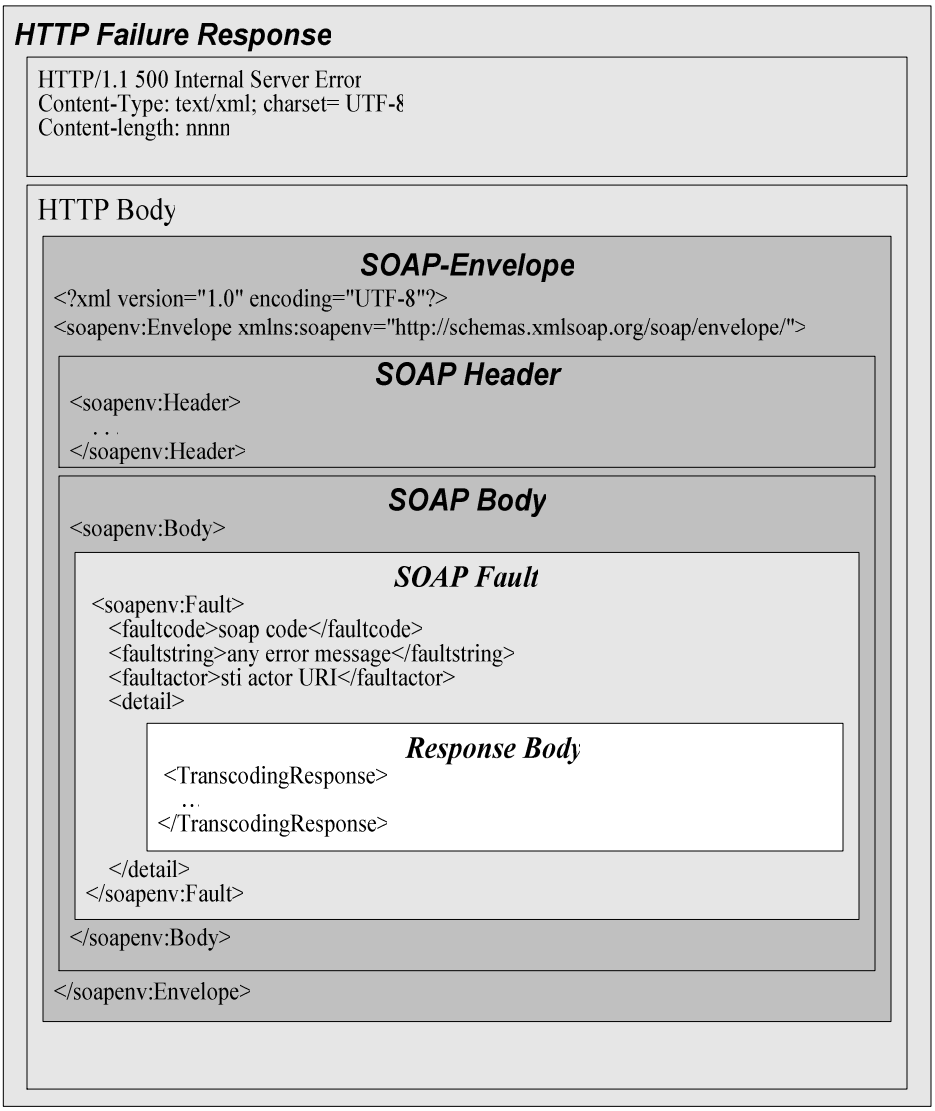

Figure 11 A Failed Transcoding Response

5.3.2.1 HTTP Response Information Associated with Failed Transcoding Responses

Since STI is a basic web service using SOAP messages over HTTP protocol to send requests and return responses asynchronously, the error handling mechanism in the response at the HTTP level must be compliant with the requirements described in the Web Services Interoperability Organization documents:

- "Basic Profile Version 1.0" as referred in [OMA MWS Guidelines], section 4
- "Attachments Profile 1.0" [WS-I Attachment Profile 1.0], section 3
- "Simple SOAP Binding Profile Version 1.0" [WS-I SOAP Binding Profile 1.0], section 3

All these documents are defined by the Web Services-Interoperability organization [WS-I]. The use of these documents is a recommendation of the OMA-MWS group.

As mentioned in section 5.3.2, a SOAP message should be included in the HTTP response of a failed Transcoding Response. The HTTP headers and header values that SHOULD or MUST be returned with SOAP (error) messages are defined in the Web Services Interoperability Organization documents listed above.

The Transcoding Platform SHALL return a "500 Internal Server Error" HTTP status code if the response message contains a SOAP Fault.. However, the web infrastructure MAY respond with other client error codes or server error code according to the scenarios described in the Web Services Interoperability Organization document s listed above. This is beyond the control of the Transcoding Platform. Following is an example of an HTTP/SOAP response (with an error 500 status code) that contains a SOAP (error) message that does not contain any attachment:

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset=UTF-8
Content-length:  nnnn
. . .

SOAP Envelope including the SOAP Fault within a SOAP Body comes here (see sections 5.3.2.2 and 5.3.3)
```

5.3.2.2 SOAP Message Information Associated with Failed Transcoding Responses

The XML representation of a SOAP (error) message has constraints as defined in the Web Services Interoperability Organization documents listed in section 5.3.2.1.

The response MAY contain references to either external content elements or self-contained content elements.

The same differences between a Transcoding Request with contained content elements, and without contained content elements, as discussed for the Transcoding Request (section 5.2.1) and successful Transcoding Response, apply for a failure Transcoding Response.

Figure 11 shows an example of the structure for a failed Transcoding Response. The SOAP-Envelope is returned as part of the HTTP response main body when no attachments are provided with the response or as part of the HTTP response first body part when some attachments are provided with the response.

The data specific to a failed Transcoding Response SHOULD be contained under the <detail> element of the SOAP Fault element of a SOAP Envelope. For more details about the SOAP Fault element, refer to the SOAP guidelines as defined in [OMA MWS Guidelines], section 6.3.

5.3.2.3 SOAP Fault Body Usage

Please see the Web Services Interoperability Organization document "Basic Profile Version 1.0" as referred in [OMA MWS Guidelines] and "Basic Profile Version 1.1" [WS-I Basic Profile 1.1] for details about how to use the SOAP Fault body.

Possible fault codes for STI:

soapenv:VersionMismatch

soapenv:MustUnderstand

soapenv:Client

soapenv:Server

Possible fault strings for STI:

Any message that describes the problem. It could be for example the same message returned within the *mainReturnResult* parameter in the *TranscodingResponse* structure.

Possible fault actor for STI:

A URI to represent the Transcoding Platform (see example below).

Content of the *detail* element:

The *detail* element in the SOAP Fault Body SHOULD contain an instance of the *TranscodingResponse* structure.

Example:

```
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' >
  <soap:Header>
        . . .
  </soap:Header>
  <soap:Body>
      <soap:Fault xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' >
        <faultcode>soap:Client</faultcode>
        <faultstring>Any error message.</faultstring>
        <faultactor>http://www.TranscodingPlatformCompany.com/sti/v1_0</faultactor>
        <detail>
            <sti_xsd:TranscodingResponse
                 xmlns:sti_xsd='http://www.openmobilealliance/schema/sti/v1_0' >
                <originatorID>originator name</originatorID>
                <operationID>101<operationID>
                . . .
            </sti_xsd:TranscodingResponse>
        </detail>
      </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

5.3.2.4 Transcoding Job Failure

The result of each Transcoding Job is returned using one instance of the *jobResult* structure per requested Transcoding Job. A failed Transcoding Job does not mean that the whole operation failed. A transcoding operation SHALL still be successful (with warnings) even if some of its Transcoding Jobs have failed. In case all the Transcoding Jobs within a Transcoding Request failed, the whole operation SHALL fail. In such a case, the Job Result of each failed Transcoding Job SHALL reflect the specific Transcoding Job's reason of failure.

5.3.3 Transcoding Response –Detailed Structure

The SOAP-based Response Body SHALL be structured as shown by the following table. This structure reflects the content of the XML Schema (.xsd) which can be found at [STI XSD]. In case of a conflict between the structures presented in this document (tables or UML) and the [STI XSD], implementations SHOULD follow the [STI XSD].

Note: The *additionalReturnResults* structure in the table below (*TranscodingResponse* and *jobResult* structures) allows the Transcoding Platform to return multiple error or warning codes and strings per request.
Text written in **bold** indicates a structure, which is defined further down in the table or in another table in this specification.

Table 8: Transcoding Response - Detailed Structure

No.	Parent	Name	Type	Units / Possible Values	Mandatory / Optional	Description
1.		**TranscodingResponse**			Conditional (0..1)	The response details. This element MUST exist if the operation was not erroneous. This element SHOULD exist in case of erroneous operations.
2.	TranscodingResponse	originatorID	Token		Conditional (0..1)	The unique originator ID that was received from the Application Platform in the request. This element SHOULD exist if *originatorID* in the TranscodingRequest was readable.
3.	TranscodingResponse	operationID	Token		Conditional (0..1)	The unique ID that was received from the Application Platform in the request. This element SHOULD exist if *operationID* in the TranscodingRequest was readable.
4.	TranscodingResponse	**mainReturnResult**			Mandatory (1..1)	The return result of the operation.
5.	TranscodingResponse	**additionalReturnResults**			Optional (0..1)	In addition to the *mainReturnResult*, a Transcoding Response can contain additional return results.
6.	TranscodingResponse	totalDuration	nonNegativeInt	In milliseconds	Mandatory (1..1)	The time it took the Transcoding Platform to perform the entire transcoding operation in milliseconds. This number is the duration of the whole operation from the time it reached the Transcoding Platform to the time the response was sent. It may be different than the sum of durations of all the jobs (either a higher value, reflecting the overall operation overhead, or a lower value

						if multiple jobs were processed in parallel).
7.	TranscodingResponse	**jobResult**			Conditional (0.. n)	The Job Results. May be more than one (one per Transcoding Job) This element MUST exist if the *mainReturnResult* of the *TranscodingResponse* is not an error.
8.	TranscodingResponse	**extensionData** (see ***Table 7***)			Optional (0..1)	Proprietary extension data.
9.	mainReturnResult	returnCode	Integer		Mandatory (1..1)	Code can be of type info, success, warning, client error, or server error. See section 5.3.4
10.	mainReturnResult	returnString	String		Optional (0..1)	Message string associated with the return code.
11.	**additionalReturnResults**	**returnResult**			Optional (0..n)	The additional return results.
12.	returnResult	returnCode	Integer		Mandatory (1..1)	Code can be of type info, success, warning, client error, or server error. See section 5.3.4
13.	returnResult	returnString	String		Optional (0..1)	Message string associated with the return code.
14.	jobResult	jobID	Token		Mandatory (1..1)	A unique ID that is given by the Application Platform to uniquely identify the Transcoding Job.
15.	jobResult	**extensionData** (see ***Table 7***)			Optional (0..1)	Proprietary extension data.
16.	jobResult	**mainReturnResult**			Mandatory (1..1)	The return result of the Transcoding Job.
17.	jobResult	**additionalReturnResults**			Optional (0..1)	In addition to the *mainReturnResult*, a Job Result can contain additional return results.
18.	jobResult	**adaptationsPerformed**			Optional (0..1)	The adaptation classes associated with the actual Content Adaptation performed.
19.	jobResult	duration	nonNegativeInt	In milliseconds	Optional (0..1)	The time it took the Transcoding Platform to perform the job.
20.	jobResult	**output**			Conditional (0..1)	Details about the result target. This element MUST exist if the *mainReturnResult* of the job is not an error.
21.	**adaptationsPerformed**	adaptationPerformed	Token		Optional (0..n)	An adaptation class associated with the actual Content Adaptation performed. The Transcoding Platform SHALL return the list of all adaptation classes present in the Transcoding Request that were encountered in the Transcoding process, each in a

						separate *adaptationPerformed* element.
22.	output	contentType	Token	According to the content types naming (see section **Error! Reference source not found.**)	Mandatory (1..1)	The content type of the resulting output.
23.	output	**contentTypeParams** (see Table 7)			Optional (0..1)	The content type parameters associated with the content type of the output.
24.	output	location	anyURI	URI	Mandatory (1..1)	For external outputs: The full-path (URI) to an external storage including the file name of the converted resource. This name MAY include a file extension proper to the transcoded content. The *location* SHALL be composed of the *path* and *name* specified in the *target*'s *externalLocation* parameter. The Transcoding Platform SHOULD add an appropriate extension to the name of the output media, if such an extension exists. In case the *name* parameter in the *target*'s *externalLocation* parameter was not specified, the name of the result media SHOULD be the name of the source media, with an appropriate extension (see ***Table 1***). For outputs included in the response this parameter is the Content ID (cid) of the attachment [RFC2392].
25.	output	mediaSize	nonNegativeLong	In bytes.	Optional (0..1)	The actual size of the media.
26.	output	**extensionData** (see ***Table 7***)			Optional (0..1)	Proprietary extension data.

5.3.4 Return Codes

When returning a code, if a situation matches a predefined code in the tables below, the Transcoding Platform SHALL use the following info, success, warning, client error and server error return codes in a *TranscodingResponse* or a *jobResult* element. When a warning code is returned, this means that the operation was executed, but with some restriction(s) and/or additional information, as defined by the return code and its message.
Additional details about the return code SHOULD also be provided as part of the *returnString* to give more information to the Application Platform.

In case there is more than one return code applicable the Transcoding Platform SHALL return at least one return code using the *mainReturnResult* element and MAY return more relevant return codes using the *additionalReturnResults* element.

The return codes defined below are 4 digit integers. The first digit of the return code defines the classification level of the response. The classification of the return codes is as follows:

- 1000-1999: Info codes.
- 2000: Success code.
- 2001-2999: Warning codes (success but on some conditions).
- 4000-4999: Client error codes (e.g.: parsing error, invalid parameter value, etc.).
- 5000-5999: Server error codes (e.g.: unsupported parameter, internal error, etc.).

Note that the upper half of each range will be reserved for proprietary codes. For example, for info codes, 1000 to 1499 are reserved for STI specification and 1500 to 1999 are reserved for proprietary codes.
In case an Application Platform does not understand a return code it should treat the return code as being equivalent to the x000 status code of that class. For example, if the Application Platform receives an unrecognized return code of 4678, it can safely assume that there was a client error and treat the response as if it contained a 4000 return code.

In the following tables, the first column contains the possible result codes. The second column contains the name of the structures which typically returns the result code (TR = Transcoding Response, JR = Job Result). The third column contains the message associated to the result code. These info messages are only recommendations - the Transcoding Platform MAY use different descriptive strings if it finds appropriate. And finally, the fourth column contains some additional information and examples about the result code.

Table 9: Informational Codes (1000-1999)

Result Code	Ret. By	Info Message	Additional Info
1001	JR	Info – Result content saved to an external location.	The result content was saved to an external location as specified in the transcoding job of the request.
1002	JR	Info – Default profile used.	No profile ID specified in the request and in the transcoding job, a default profile configured on the Transcoding Platform was used instead.
1003	JR	Info – No transcoding performed - original content returned.	The content returned by the Transcoding Platform was not modified during the operation since no transcoding were necessary.
1004	JR	Info – The transcoding did not result in any content.	The Transcoding Platform did not return any content with the response of a transcoding job. There are many reasons why a Transcoding Platform MAY decide not to return any content along with the job result: - No modification performed on the source content while transcoding. - Content was resized to zero. - Any error occurred while transcoding.(this code would be in addition to the error code) - No source content to transcode was provided along with the request or was accessible from the Transcoding Platform, (this code would be in addition to the error code).
1005-1499		STI – reserved for future use	
1500-1999		Other– Reserved for any other non-defined (or proprietary) info message.	

Table 10: Success / Warning Codes (2000-2999)

Result Code	Ret. By	Success/Warning Message	Additional Info
2000	TR	Success – Successful result.	A Transcoding Request or a Transcoding Job was

	JR		performed successfully. Some additional info or warning result codes/messages MAY be added to the response.
2001	TR	Warning – One or more transcoding jobs failed.	The operation was successful but some of its transcoding jobs failed (at least one transcoding job succeeded).
2002	TR	Warning – Multiple Transcoding Jobs not supported, only one job processed successfully.	The Transcoding Platform does not support multiple transcoding jobs per operation. Only one job performed successfully.
2003	TR JR	Warning – Unsupported parameter ignored, not used by the Transcoding Platform.	A given request parameter which is not supported by the Transcoding Platform was ignored, but the whole operation or the transcoding job was performed successfully.
2004	TR JR	Warning – Unsupported parameter value ignored.	A valid request parameter value which is not supported by the Transcoding Platform was ignored, transcoding was performed.
2005	TR JR	Warning – Unknown parameter value ignored.	A given request parameter value which is unknown to the Transcoding Platform was ignored, transcoding was performed.
2006	JR	Warning – Content truncated.	One or more contents had to be truncated (e.g., to meet the sizeLimit, or to meet the device's capabilities).
2007	JR	Warning – Content removed.	One or more contents were removed (e.g., to meet the sizeLimit, or to meet the device's capabilities).
2008	JR	Warning – Unable to transcode DRM protected content.	DRM protection was found on one or more elements. No transcoding performed on these elements.
2009	JR	Warning – DRM protected content transcoded.	Even though the content was DRM protected, it was transcoded by the Transcoding Platform.
2010	JR	Warning – Incompatibility between transcodingParams and profileID ignored.	One or more requested transcodingParams element not compatible with profile associated with profileID, transcoding performed as requested
2011	JR	Warning – Neither profileID nor transcodingParams specified	Neither profileID nor transcodingParams were specified in the request, default profile was used for transcoding.
2012-2499		STI – reserved for future use	
2500-2999		Other– Reserved for any other non-defined (or proprietary) warning result.	

Table 11: Client Error codes (4000-4999)

Result code	Ret. By	Error Message	Additional Info
4000	TR JR	Client Error	Generic client error
4001	TR	Client Error – Parsing error: Invalid STI Request (in SOAP message body)	The format of the SOAP Message received is valid but the format of the STI Transcoding Request in the message body appears to be invalid. The Transcoding Platform is unable to process the request.
4002	TR JR	Client Error – Unknown parameter value.	The value of a given parameter is unknown by the Transcoding Platform which is unable to continue processing the request.
4003	TR	Client Error – Unauthorized request.	When a request is not authorized to be processed based on one of the request parameters, like the originatorID, applicationType, , etc.

4004	TR JR	Client Error – Unable to get the specified internal or external resource: Invalid URI	The given source location URI (representing an internal or external location) is invalid and cannot be resolved. The resource could be an internal (attached) or external input data (e.g. media, policy reference, profile, etc).
4005	TR JR	Client Error – Unable to get the specified internal or external resource: Location forbidden.	The Transcoding Platform does not have the permission to access the resource at the given location. Note: A forbidden access usually occurs for external resources. The resource could be an internal (attached) or external input data (e.g. media, policy reference, profile, etc)..
4006	TR JR	Client Error – Unable to get the specified internal or external resource: Location not found.	No resource found at the given location. The resource could be an internal (attached) or external input data (e.g. media, policy reference, profile, etc).
4007	TR JR	Client Error – Error while reading the specified internal or external resource: IO problem.	A resource referenced in the request has been found but an error occurred while reading it. The resource could be an internal (attached) or external input data (e.g. media, policy reference, profile, etc)
4008	JR	Client Error – Failed to save the resource to the specified target external location: Invalid URI	The given target location URI (representing an external location) is invalid and cannot be resolved. Unable to save the resource (e.g. transcoded content).
4009	JR	Client Error – Failed to save the resource to the specified target external location: Location forbidden.	The given target location URI (representing an external location) cannot be accessed. The Transcoding Platform does not have the permission to access the given URI. Unable to save the resource (e.g. transcoded content).
4010	JR	Client Error – Failed to save the resource to the specified target external location: Location not found.	The given target location URI (representing an external location) cannot be found. Unable to save the resource (e.g. transcoded content).
4011	JR	Client Error – Failed to save the resource to the specified target external location: IO Problem	The given target location URI (representing an external location) is valid but an IO error occurred while saving the resource.
4012	JR	Client Error - Adaptation not Allowed	The Transcoding Platform is unable to perform the transcoding within the Adaptation Class restrictions.
4013	JR	Client Error – Non Unique jobID	Two or more transcoding jobs within the same request have the same jobID.
4014	JR	Client Error – Neither profileID nor transcodingParams specified	Neither profileID nor transcodingParams were specified in the request, and no default profile was used. No transcoding was performed.
4015	JR	Client Error – Insufficient transcoding parameters provided.	Some transcoding parameters (either profileID or transcodingParams) were provided, but the information is insufficient.
4016	TR JR	Client Error - Error reading a specified internal or external resource.	The Transcoding Platform encountered an error when reading the specified internal or external resource, and the error could not be mapped to one of the more specific return codes 4004-4007.
4017	JR	Client Error – Error when accessing a specified external location for writing.	The Transcoding Platform encountered an error when saving to the specified external location, and the error could not be mapped to one of the more specific return codes 4008-4011.
4018	TR JR	Client Error - Error when processing a specified internal or external resource.	The Transcoding Platform encountered an error when processing the specified internal or external resource

			(e.g. the source media was read successfully but it was corrupt).
4019-4499		STI – reserved for future use	
4500-4999		Other – Reserved for any other non-defined (or proprietary) client error result.	

Table 12: Server Error Codes (5000-5999)

Result code	Ret. by	Error string	Additional Info
5000	TR JR	Server Error – Internal Server Error.	General error code to represent any error caused internally by the Transcoding Platform.
5001	TR JR	Server Error – Unsupported parameter.	A given request parameter which is not supported by the Transcoding Platform forced it to stop the processing of the request or the transcoding job.
5002	TR JR	Server Error – Unsupported parameter value.	A given request parameter value which is not supported by the Transcoding Platform forced it to stop the processing of the request or the transcoding job.
5003	JR	Server Error – Transcoding service temporary unavailable.	A transcoding job cannot be performed because the Transcoding Platform is temporary unavailable for any reason (could be busy, in maintenance state, etc.)
5004	TR JR	Server Error – Timeout.	The operation or a transcoding job was stopped because of an internal system timeout.
5005	TR	Server Error – STI Version not supported.	The provided request is for an STI version which is not supported by the Transcoding Platform. *Note that this error code does not stand for an invalid STI version since that should normally be returned as a parsing error.*
5006	JR	Server Error – Unable to perform transcoding.	The Transcoding Platform is unable to perform the requested transcoding (e.g., the device capabilities/profile show that it only supports image/gif and the Transcoding Platform does not support this format)
5007	JR	Server Error – Cannot meet sizeLimit.	The Transcoding Platform is unable to meet the sizeLimit.
5008	JR	Server Error – DRM content – No transcoding performed.	DRM protection was found on one or more elements. No transcoding performed.
5009	TR	Server Error – All the transcoding jobs failed.	All the transcoding jobs in the Transcoding Request failed.
5010	TR JR	Server Error – License prohibits the request	When a licensing agreement would be exceeded.
5011	TR	Server Error – Resource Limit Exceeded	Resource could be memory, CPU, etc.
5012	TR	Server Error – Multiple Transcoding Jobs not supported.	The Transcoding Platform does not support multiple transcoding jobs per operation. None of the transcoding job was performed.
5013	JR	Server Error – Input media not supported	The Transcoding Platform does not support the input media and cannot open it.
5014-5499		STI – reserved for future use	
5500-5999		Other – Reserved for any other non-defined (or proprietary) server error result.	

6. Charging (Informative)

STI 1.0 does not include any specific charging parameters such as money amount, etc. The Transcoding Platform will however return basic information such as operation ID, return code, total duration, and file size which can be used by the requesting application to perform rating and charging operations. Note that the Application Platforms and the Transcoding Platform can either be within the same network (i.e. trusted nodes) or in different networks (untrusted nodes).

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version –or- No previous version within OMA

A.2 Draft/Candidate Version 1.0 History

Document Identifier	Date	Sections	Description
Draft Versions OMA-STI-V1_0	15 Jan 2004	All	First Draft
	25 Jan 2004	All	
	27 Jan 2004	All	
	25 Feb 2004	All	
	10 Mar 2004	All	
	22 Apr 2004	All	
	17 May 2004	5.1.3.1, 5.1.4, 5.2.2.2, 5.2.6.2, 5.3.2.2, 5.5, Appendix F	Changed parameters' names and locations, made some clarifications and corrections, removed section 5.5 (service discovery).
	11 Jul 2004	5.2.3, 5.2.4, 52.5, 5.2.6.2, 5.3, Appendix C, Appendix E, Appendix H, UML Diagrams	Sections 5.2.3, 5.2.4 rewritten (media types, content types and codecs), 5.3 revised – failureResponse merged with TransactionResponse, added content type examples, added SCR tables, new UMLs, new xsd.
	12 Aug 2004	5.1.2, Appendix H, All tables	Added reference to OWSER document, Numbered tables, added applicationSizeLimit, changed originatorID comment, some fixes to SCR tables.
	23 Sep 2004	3, 5.1.3.4, 5.1.5, 5.5, All tables, UML diagrams	Added URI to abbreviations, Changed multipart description, Added explanations about ProfileID and TranscodingParams hierarchy, Added section 5.5, Updated all tables and UML diagrams and other minor clerical changes.
	1 0ct 2004	Appendices	Removed xsd schema, which will be uploaded in a separate document, removed examples until they are appropriately updated.
	18 0ct 2004	5.1.3, 5.1.6, 5.2.5, 5.3, Table 1, Tables 6-12, Appendix D.	Added Data Types section, added sections about adaptation classes and size limit hierarchy, revision of Transformations section, revision of Response Transaction section, return codes and errors codes. Added appendix D for adaptation classes examples.
	11 Nov 2004	5.1.2, 5.1.5.1, 5.1.6.4, Figure 7, 5.2.1.1, 5.2.1.2, 5.2.2.1, 5.3.1.1, 5.3.3, 6	Some changes regarding protocols to referenced resources, changed externalLocation structure in Target, small changes according to MWS recommendations, removed section 6 about DRM.
	28 Nov 2004		Made public.
	8 Dec 2004	2.1, 3.2, 5.2, 5.3, tables 1, 7, 8, SCR tables, UML diagrams, Appendices	Removed redundant references and abbreviations, renamed RequestTransaction and ResponseTransaction to TranscodingRequest and TranscodingResponse, made some response fields Conditional, changes in some return codes, added contentType in output, reorder of the parameters in the tables, made some very slight changes in the parameters, replaced SCR tables, updated UML diagrams, added appendix B.
	11 Jan 2005	5.1.3, 5.1.5, tables 7,8,10, appendix f, g, h.	Added types nonNegativeInt, nonNegativeLong, unboundedLong, unboundedInt. Added examples appendixes, new SCR tables (R05).
	27 Jan 2005	2, 5.1.6.6, 5.1.4.3, 5.2.2.1, 5.2.4, 5.3, tables 6,8, figure 7, appendixes B,C	Codec table, codec parameters table, new overall UML diagram and detailed UML diagrams, removed Policy appendix. Editorial changes.
	9 Feb 2005	5.1.4, 5.3, 5.2.5, tables 3, 6, 7, appendixes	Editorial changes, new parameter in Image and VideoVisual object, new section 5.1.4, moved SCR tables appendix, new appendices H, I.

Document Identifier	Date	Sections	Description
	16 May 2005	2.1, 3.2, 3.3, 5.3, Figures 7, 9, 12, 13, 15, Tables 1, 3, 5, 6, 7, 11, 12. Appendix B, I, J.	Consistency review changes: Fixed references, added some contentType and codec parameters, moved applicationSizeLimit from transcodingParams to TranscodingRequest and Target, some changes in 4xxx return codes, clarified section 5.3 about TranscodingResponse, fixed some SCR entries, added appendix I.
Candidate Versions: OMA-TS-STI-V1_0	07 Jun 2005	n/a	Status changed to Candidate by TP TP ref #OMA-TP-2005-0181-STI-V1_0-for-Candidate-approval

Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [IOPPROC].

B.1 Client Conformance Requirements

The table below enumerates the client conformance requirements. A Client being any Application Platform able to use the features and services offered by the Transcoding Platform

Item	Function	Reference	Status	Requirements
STI-CLI-001	SOAP 1.1 protocol support	5.1.2	M	Support of SOAP 1.1 according to [OMA MWS Guidelines] AND STI-CLI-002 AND STI-CLI-003 AND STI-CLI-004 AND STI-CLI-005 AND STI-CLI-006
STI-CLI-002	SOAP request sent over HTTP(S)	5.1.2	M	Support of HTTP(S) according to [OMA MWS Guidelines]
STI-CLI-003	SOAP request sent as body of a HTTP(S) POST	5.1.2	M	
STI-CLI-004	SOAP request containing a single SOAP envelope	5.1.2	M	
STI-CLI-005	Attachment structure conformant with the definition of SOAP with Attachments [OMA MWS Guidelines]	5.1.2	M	
STI-CLI-006	Use of the "start parameter" within SOAP protocol	5.1.2	M	
STI-CLI-007	Use of the canonical lexical representation of datatypes defined within [XML Schema Part 2: Datatypes]	5.1.3	M	
STI-CLI-008	Follow case sensitivity guidelines	5.1.4	M	
STI-CLI-009	SOAP Header	5.1.5.1	O	
STI-CLI-010	Content files included in the HTTP(S) POST message but outside the SOAP	5.1.5.1	O	

	message			
STI-CLI-011	Support of at least one Transcoding Job within a single Transcoding Request	5.1.5.1	M	
STI-CLI-012	Support of multiple Transcoding Jobs within a single Transcoding Request	5.1.5.1	O	
STI-CLI-013	Support of Transcoding Job structure	5.1.5.1	M	STI-CLI-014
STI-CLI-014	Each Transcoding Job containing source element	5.1.5.1	M	STI-CLI-051
STI-CLI-015	Each Transcoding Job containing target element	5.1.5.1	O	STI-CLI-052
STI-CLI-016	Source element containing contentType and location	5.1.5.3	M	
STI-CLI-017	Support of image transcoding	5.2.3	O	
STI-CLI-018	Support of audio transcoding	5.2.3	O	
STI-CLI-019	Support of video transcoding	5.2.3	O	
STI-CLI-020	Support of text transcoding	5.2.3	O	
STI-CLI-021	Support of multipart transcoding	5.1.5.4	O	STI-CLI-022 AND STI-CLI-023
STI-CLI-022	Multipart transferred as a MIME-Multipart content item	5.1.5.4	O	See [MIME]
STI-CLI-023	Multipart transferred in the WAP binary form of the MIME Multipart using appropriate MIME types	5.1.5.4	O	See [WAPWSP]
STI-CLI-024	First content attachment contains SOAP-based Request Body (SOAP envelope)	5.2.1.2	M	
STI-CLI-025	Content data referenced from within the SOAP Request body	5.1.6	M	STI-CLI-026 OR STI-CLI-034
STI-CLI-026	Self contained content data	5.1.6	O	STI-CLI-027 AND STI-CLI-033 AND STI-CLI-037
STI-CLI-027	Content referenced as attachment	5.1.6.2, 5.2.1.2	O	STI-CLI-028 AND STI-CLI-029 AND

				STI-CLI-030 AND STI-CLI-031 AND STI-CLI-032
STI-CLI-028	Attachments sent along with the Request as MIME parts according to the SOAP with Attachments definition in [OMA MWS Guidelines]	5.1.6.2	O	
STI-CLI-029	Each content attachment identified by its MIME Content ID	5.1.6.2	O	
STI-CLI-030	Content referenced by its Content ID within the SOAP Request Body	5.1.6.2	O	
STI-CLI-031	Each content attachment contains its own header using MIME	5.2.1.2	O	
STI-CLI-032	Each content attachment header contains the unique ID, content-type and content-transfer-encoding	5.2.1.2	O	[RFC2392] for the content ID
STI-CLI-033	HTTP headers: Target host, content length (total content of the request), content type, SOAP action and boundary string	5.2.1.2	O	
STI-CLI-034	References to external content elements	5.1.6.1, 5.2.1.1	O	STI-CLI-035 AND STI-CLI-036 AND STI-CLI-037
STI-CLI-035	URI using HTTP and HTTPS	5.1.6.1	O	
STI-CLI-036	HTTP headers: Target host, content length, content type and SOAP action	5.2.1.1	O	
STI-CLI-037	Charset-parameter present in the SOAP envelope's content type	5.2.1.1	M	STI-CLI-038 OR STI-CLI-039
STI-CLI-038	Support UTF-8	5.2.1.1	O	
STI-CLI-039	Support UTF-16	5.2.1.1	O	
STI-CLI-040	Specification of transcoding instructions	5.1.7	M	STI-CLI-041 OR STI-CLI-042
STI-CLI-041	Support of predefined Profiles	5.1.7.1	O	STI-CLI-043
STI-CLI-042	Support of explicit transcoding parameters	5.1.7.2	O	STI-CLI-043

STI-CLI-043	profileID and explicit transcoding parameters hierarchy scheme	5.1.7.3	O	
STI-CLI-044	Support of Policy parameters	5.1.7.4	O	STI-CLI-045
STI-CLI-045	Use of predefined Policy directed by an URI (policyRef)	5.1.7.4	O	
STI-CLI-046	Support of Adaptation Classes	5.1.7.5	O	
STI-CLI-047	Transcoding Request generation	5.2.1, 5.2.2.1	M	
STI-CLI-048	originatorID: Unique ID identifying the Application Platform	5.2.2.1	M	
STI-CLI-049	operationID: Unique ID identifying the request	5.2.2.1	M	
STI-CLI-050	jobID: Unique ID identifying the Transcoding Job within the request	5.2.2.1	M	
STI-CLI-051	Support of source element structure.	5.2.2.1	M	STI-CLI-053 OR STI-CLI-054
STI-CLI-052	Support of target element structure.	5.2.2.1	O	STI-CLI-055 AND STI-CLI-056
STI-CLI-053	Location if contents included: Content ID (cid) of the attachment within the request as in [RFC2392]	5.2.2.1	O	
STI-CLI-054	Location if contents referenced: Full-path (URI) to an external storage	5.2.2.1	O	
STI-CLI-055	Support of externalLocation structure to indicate the location of a target content	5.2.2.1	O	
STI-CLI-056	Target element specifying desired transcoding information	5.1.4.3, 5.2.2.1	O	STI-CLI-057 OR STI-CLI-058
STI-CLI-057	Target element specifying explicit transcoding parameters	5.1.4.3, 5.2.2.1	O	
STI-CLI-058	Target element specifying Profile information	5.1.4.3, 5.2.2.1	O	
STI-CLI-059	Support of standard transformations	5.2.5	O	
STI-CLI-060	Support the receiving of Transcoding Response	5.1.1, 5.3	M	
STI-CLI-061	Support of image content in Transcoding Requests	5.2.3	O	
STI-CLI-062	Support of audio content in Transcoding Requests	5.2.3	O	

STI-CLI-063	Support of video content in Transcoding Requests	5.2.3	O	
STI-CLI-064	Support of text content in Transcoding Requests	5.2.3	O	
STI-CLI-065	Support of multipart content in Transcoding Requests	5.1.5.4	O	

B.2 Server Conformance Requirements

The table below enumerates the server conformance requirements. A Server being any Transcoding Platform able to offer transcoding services to the Applications Platforms.

Item	Function	Reference	Status	Requirements
STI-SERVER-001	SOAP 1.1 protocol support	5.1.2	M	Support of SOAP 1.1 according to [OMA MWS Guidelines] STI-SERVER-002 AND STI-SERVER-003 AND STI-SERVER-004 AND STI-SERVER-005 AND STI-SERVER-006
STI-SERVER-002	SOAP response sent over HTTP(S)	5.1.2	M	Support of HTTP(S) according to [OMA MWS Guidelines]
STI-SERVER-003	SOAP response sent as body of a HTTP(S) response	5.1.2	M	
STI-SERVER-004	SOAP response containing a single SOAP envelope	5.1.2	M	
STI-SERVER-005	Attachment structure conformant with the definition of SOAP with Attachments [OMA MWS Guidelines]	5.1.2	M	
STI-SERVER-006	Use of the "start parameter" within SOAP protocol	5.1.2	M	
STI-SERVER-007	Use of the canonical lexical representation of datatypes defined	5.1.3	M	

	within [XML Schema Par2: Datatypes]			
STI-SERVER-008	Follow case sensitivity guidelines	5.1.4	M	
STI-SERVER-009	Support of a single Job Result within a Transcoding Response	5.1.5.2	M	
STI-SERVER-010	Support of multiple Job Results within a single Transcoding Response	5.1.5.2	O	
STI-SERVER-011	Support of image content	5.2.3	O	
STI-SERVER-012	Support of audio content	5.2.3	O	
STI-SERVER-013	Support of video content	5.2.3	O	
STI-SERVER-014	Support of text content	5.2.3	O	
STI-SERVER-015	Support of multipart content	5.1.5.4	O	STI-SERVER-016 OR STI-SERVER-017
STI-SERVER-016	Multipart transferred as a MIME-Multipart content item	5.1.5.4	O	See [MIME]
STI-SERVER-017	Multipart transferred in the WAP binary form of the MIME Multipart using appropriate MIME types	5.1.5.4	O	See [WAPWSP]
STI-SERVER-018	Content data referenced from within the SOAP Request body	5.1.6	O	STI-SERVER-019 AND STI-SERVER-030
STI-SERVER-019	Content inclusion mechanisms	5.1.6	O	STI-SERVER-020 OR STI-SERVER-025
STI-SERVER-020	Self contained content data	5.1.6	O	STI-SERVER-021
STI-SERVER-021	Content referenced as attachment	5.1.6.2	O	STI-SERVER-022 AND STI-SERVER-023 AND STI-SERVER-024
STI-SERVER-022	Attachments sent along with the Response as MIME parts according to the SOAP with Attachments definition in [OMA MWS Guidelines]	5.1.6.2	O	
STI-SERVER-023	Each attachment identified by its MIME Content ID	5.1.6.2	O	

STI-SERVER-024	Content referenced by its Content ID within the SOAP Response Body	5.1.6.2	O	
STI-SERVER-025	References to external content elements	5.1.6.1	O	STI-SERVER-026
STI-SERVER-026	URI using HTTP and HTTPS	5.1.6.1	O	
STI-SERVER-027	Charset present in the SOAP envelope's content type	5.2.1.1	M	STI-SERVER-028 OR STI-SERVER-029
STI-SERVER-028	Support UTF-8	5.2.1.1	O	
STI-SERVER-029	Support UTF-16	5.2.1.1	O	
STI-SERVER-030	Incoming XML encoding pseudo parameter ignored	5.2.1.1	O	
STI-SERVER-031	Support of Adaptation Classes	5.1.7.5	O	
STI-SERVER-032	Support of size limit parameters and hierarchy: - At media level. - At transcodingJob level. - At request level - Application size limit	5.1.7.6	O	
STI-SERVER-033	Support for receiving and parsing Transcoding Request	5.1.1, 5.2	M	
STI-SERVER-034	Support of receiving transcoding instructions	5.1.7	M	STI-SERVER-035 OR STI-SERVER-036
STI-SERVER-035	Support of predefined Profiles	5.1.7.1	O	STI-SERVER-038
STI-SERVER-036	Support of explicit transcoding parameters	5.1.7.2	O	STI-SERVER-038
STI-SERVER-037	Support of Policy reference	5.1.7.4	O	
STI-SERVER-038	Hierarchy of transcoding parameters: transcodingParams (job level) profileID (job level) transcodingParams (request level) profile ID (request level)	5.1.7.3	M	
STI-SERVER-039	Support of standard Transformations	5.2.5	O	

STI-SERVER-040	Support of Target layout parameters and precedence: Presentation layout Layout inside Presentation Template Layout of the source input presentation file Default: Portrait	5.2.6.2	O	
STI-SERVER-041	Transcoding Response generation	5.1.1, 5.3	M	
STI-SERVER-042	Successful Transcoding Response HTTP headers: Success code (200), content length, content type.	5.3.1.1	M	
STI-SERVER-043	Content data referenced from within the SOAP Response body	5.1.5, 5.3.1	M	STI-SERVER-044 OR STI-SERVER-048
STI-SERVER-044	Self contained content data	5.1.5, 5.3.1.2	O	STI-SERVER-045 AND STI-SERVER-046 AND STI-SERVER-047
STI-SERVER-045	Content files included in the HTTP message but outside the SOAP message	5.1.3.1, 5.3.1.2	O	
STI-SERVER-046	Content referenced as attachment	5.1.5.2, 5.2.1.2, 5.3.1.2	O	MIME parts according to SOAP with Attachments [OMA MWS Guidelines]
STI-SERVER-047	HTTP headers: content length (total content of the response), content type and boundary string	5.2.1.2, 5.3.1.2	O	
STI-SERVER-048	References to external content elements	5.1.5, 5.3.1.2	O	STI-SERVER-049 AND STI-SERVER-050
STI-SERVER-049	URI using HTTP and HTTPS	5.1.5.1, 5.3.1.2	M	
STI-SERVER-050	HTTP headers: content length and content type	5.2.1.1, 5.3.1.2	O	
STI-SERVER-051	Include Failure Transcoding Response element	5.3.2	O	STI-SERVER-052

STI-SERVER-052	Locate the Failure Transcoding Response element inside the <detail> element of the special SOAP Fault body part	5.3.2	O	[OMA MWS Guidelines]
STI-SERVER-053	originatorID: Return the unique originatorID received in the Transcoding Request	5.3.3	M	
STI-SERVER-054	operationID: Return the unique operationID received in the Transcoding Request	5.3.3	M	
STI-SERVER-055	jobID: Return the unique ID(s) given by the Application Platform to identify the Transcoding Job(s)	5.3.3	M	
STI-SERVER-056	Result of the Transcoding Request (mainReturnResult entity)	5.3.3	M	
STI-SERVER-057	Total time used to perform the entire transcoding request (totalDuration element)	5.3.3	M	
STI-SERVER-058	Support of the output entity for successful Transcoding Job	5.3.3	M	STI-SERVER-059 AND STI-SERVER-060
STI-SERVER-059	Output location for self contained content : Content ID (cid) of the attachment within the request as in [RFC2392]	5.3.3	M	
STI-SERVER-060	Output location for externally referenced content: Full-path (URI) to an external storage	5.3.3	M	
STI-SERVER-061	Support of informational return codes for Transcoding Response and Transcoding Job	5.3.4	M	
STI-SERVER-062	Support of Transcoding Response return codes: Success and warnings	5.3.4	M	
STI-SERVER-063	Support of Transcoding Job return codes: Success and warnings	5.3.4	M	
STI-SERVER-064	Support of Transcoding Response error codes: Client error codes	5.3.4	M	
STI-SERVER-065	Support of Transcoding Response error codes: Server Error codes	5.3.4	M	
STI-SERVER-066	Support of Transcoding Job error codes: Client error codes	5.3.4	M	
STI-SERVER-067	Support of Transcoding Job error codes: Server error codes	5.3.4	M	

STI-SERVER-068	HTTP response with a 500 status code for failed Transcoding Responses (with a 4xxx or 5xxx STI error code).	5.3.2	M	STI-SERVER-069
STI-SERVER-069	HTTP response including SOAP message with the SOAP Fault element in the SOAP body in case of failure	5.3.2	M	
STI-SERVER-070	HTTP status code "500 Internal Server Error" with SOAP Fault (not necessarily triggered by STI error codes 4xxx, 5xxx)	5.3.2.1	M	

Appendix C. Detailed UML Diagrams (Normative)

C.1 Overall detailed UML Diagram

The following diagram shows a more detailed view including the Property class.

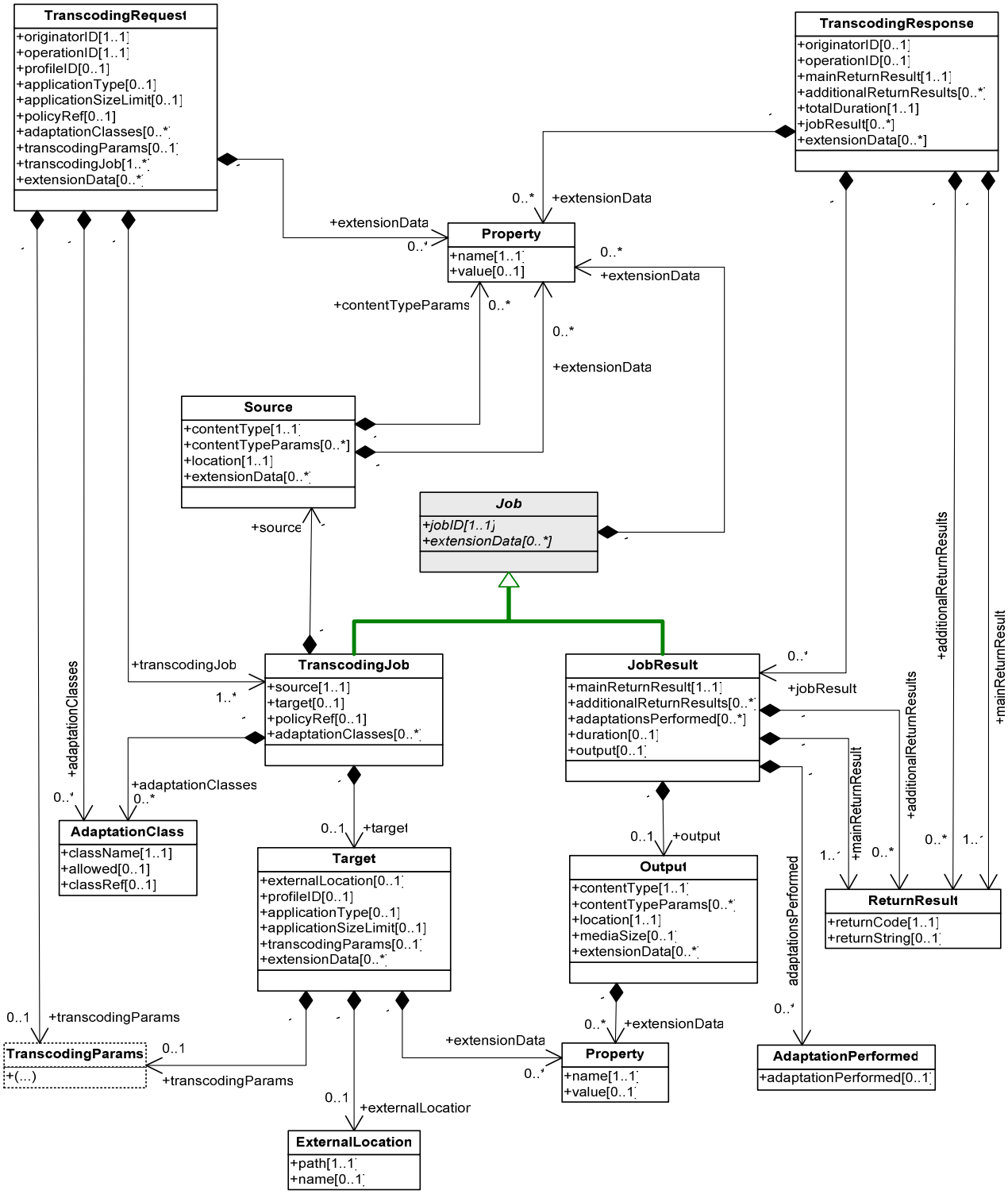

Figure 12 Detailed Overall UML Diagram

C.2 Transcoding Params detailed UML Diagram

The following diagram shows a more detailed view of the *transcodingParams* structure including the Property class and an additional Transformation class.

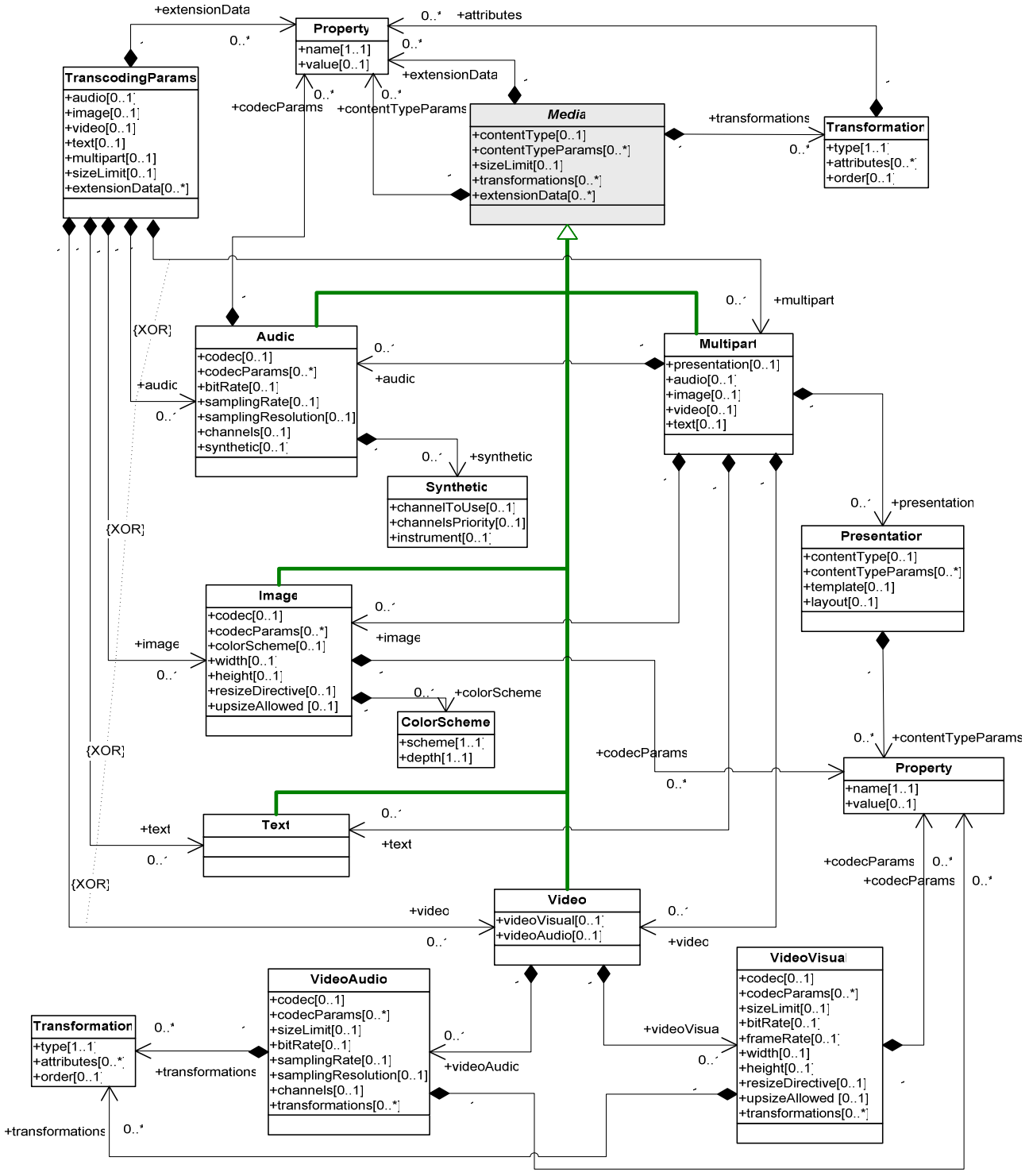

Figure 13 Detailed transcodingParams UML Diagram

C.3 transcodingParams - Media View UML Diagram

The following diagram shows a Media view of the *transcodingParams* structure. It shows that every media type inherits the attributes from the Media abstract class.

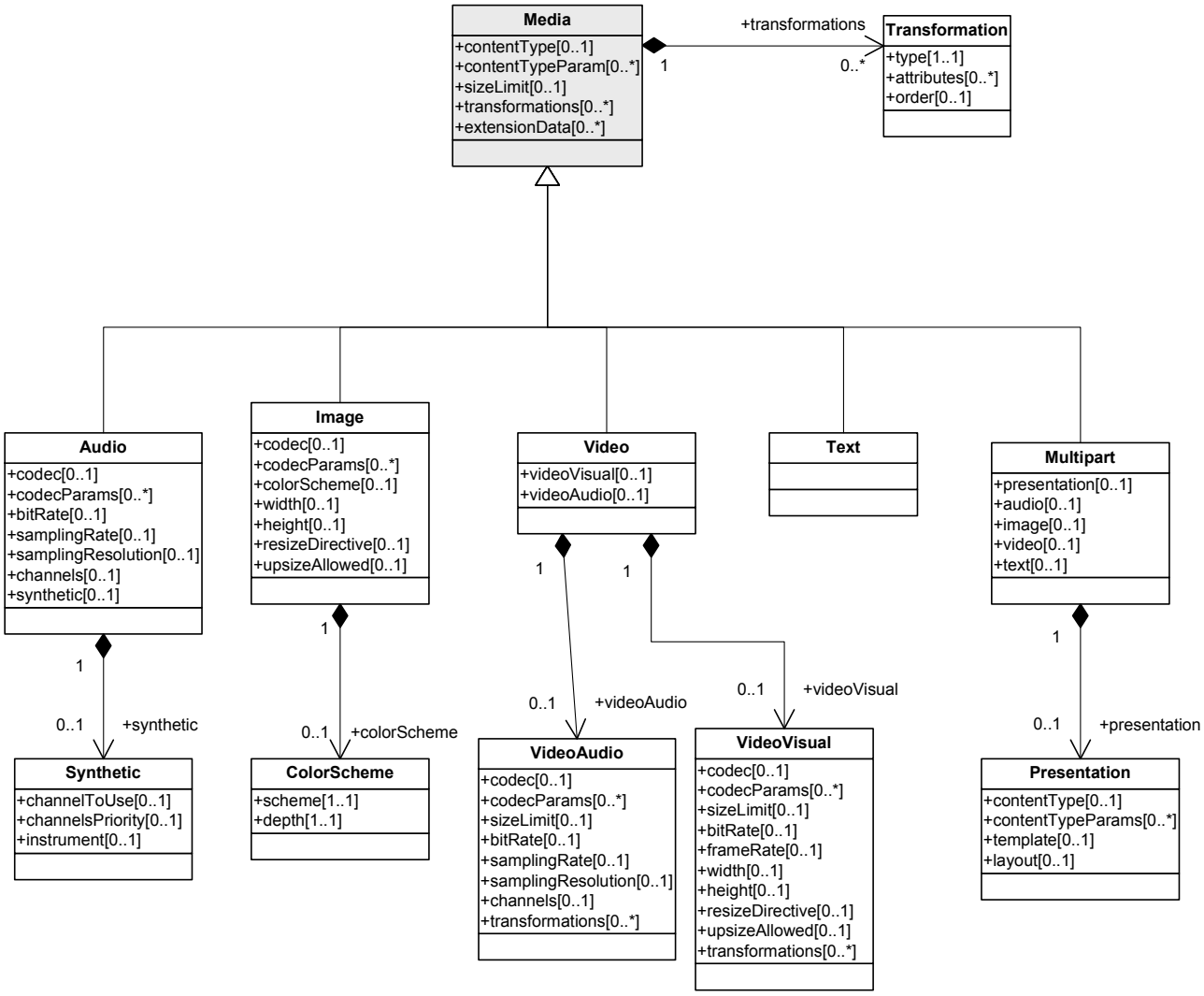

Figure 14 transcodingParams - Media View UML Diagram

C.4 transcodingParams - Multipart View UML Diagram

The following diagram shows a Multipart view of the *transcodingParams* structure. It shows that from *transcodingParams*, only one of the media types can appear (XOR), but that in a multipart, one or more media types can be specified.

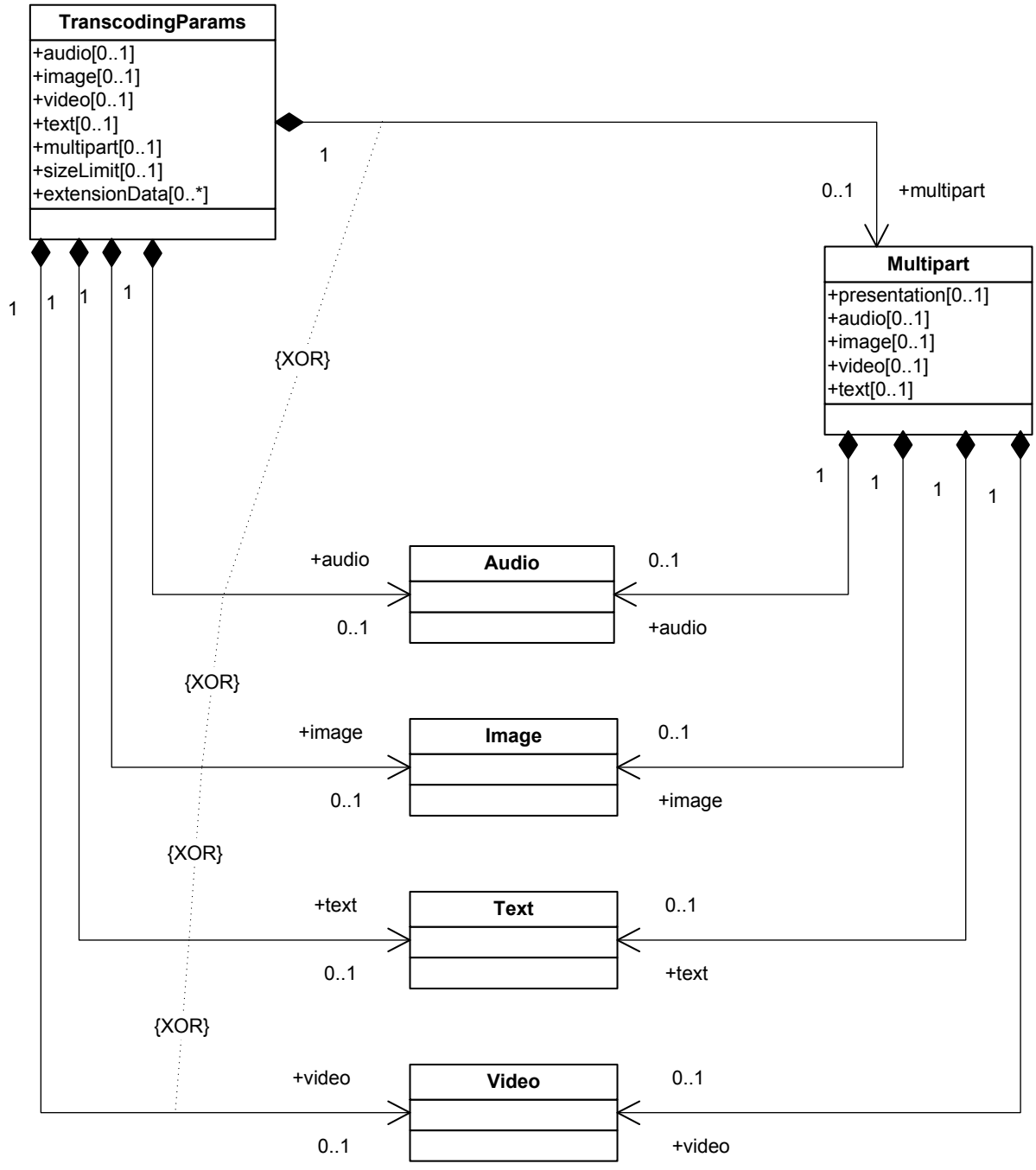

Figure 15 transcodingParams - Multipart View UML Diagram

C.5 UML Diagram for Detailed use of the Property Class

The following diagram shows that contentTypeParams is a collection of items of type Property. The name of the collection is "Properties". The attribute *extensionData* also uses the Properties collection, as well as other attributes such as *codecParams*, *attributes* (part of *transformation*). The two latter are not shown here.

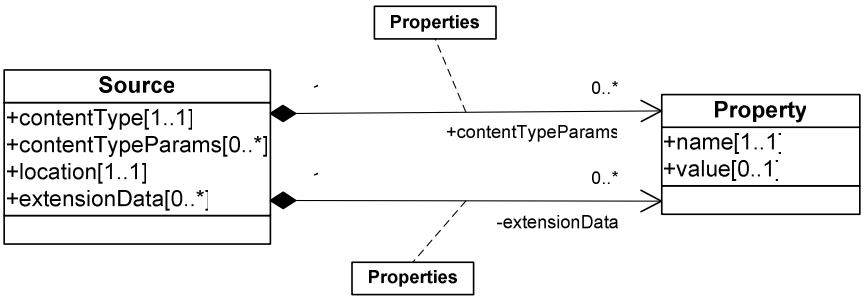

Figure 16 UML Diagram - Detailed use of the Property Class

C.6 UML Diagram for Detailed use of the Transformation Class

The following diagram shows that the *transformations* attribute of Media is a collection of items of type Transformation. The Transformation class itself has an attribute called "attributes" which uses the Properties collection.

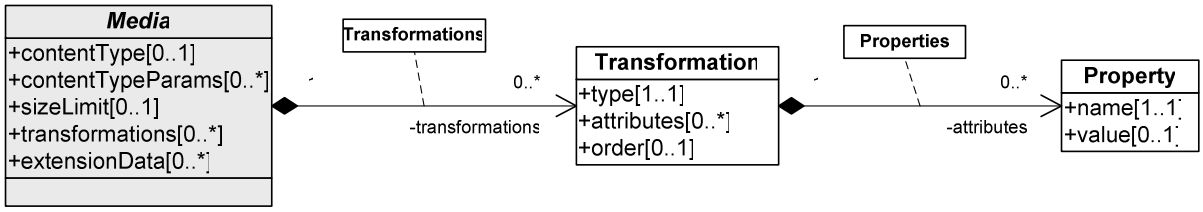

Figure 17 UML Diagram - Detailed use of the Transformation class

Appendix D. Content Type and Codec Examples (Informative)

D.1 Audio

```
<audio>
        <contentType>audio/3gpp</contentType>
        <contentTypeParams>
                <property>
                        <name>brand</name>
                        <value>3gp2</value>
                </property>
        </contentTypeParams>
        <codec>audio/amr</codec>
</audio>
```

D.2 Video

```
<video>
        <contentType>video/3gpp</contentType>
        <contentTypeParams>
                <property>
                        <name>brand</name>
                        <value>3gp6</value>
                </property>
        </contentTypeParams>
        <videoVisual>
                <codec>video/h263-2000</codec>
                <codecParams>
                        <property>
                                <name>profile</name>
                                <value>0</value>
                        </property>
                        <property>
                                <name>level</name>
                                <value>10</value>
```

```
                </property>
            </codecParams>
        </videoVisual>
</video>
```

D.3 Image

```
<image>
        <contentType>image/gif</contentType>
        <contentTypeParams>
            <property>
                <name>subset</name>
                <value>gif89a</value>
            </property>
        </contentTypeParams>
</image>
```

D.4 Text

```
<text>
        <contentType>text/plain</contentType>
        <contentTypeParams>
            <property>
                <name>charset</name>
                <value>iso-8859-1</value>
            </property>
        </contentTypeParams>
</text>
```

D.5 Multipart

```
<multipart>
        <contentType>multipart/related</contentType>
        <contentTypeParams>
```

```
            <property>
                <name>start</name>
                <value>cid:AAA.smil</value>
            </property>
            <property>
                <name>type</name>
                <value>application/smil</value>
            </property>
        </contentTypeParams>
        <presentation>
            <contentType>application/smil</contentType>
            <contentTypeParams>
                <property>
                    <name>subset</name>
                    <value>SMIL-CONF-1_2</value>
                </property>
            </contentTypeParams>
        </presentation>
</multipart>
```

Appendix E. Example of Adaptation Classes (Informative)

This section provides illustrative examples on the description and use of Adaptation Classes in the STI v1.0 interface.

E.1 MMS Major/Minor

The first example illustrates how an MMSC could describe minor and major adaptations to an STI-compliant Transcoding Platform. Both minor and major adaptations are allowed by default. In the Transcoding Request, we only present the part that relates to adaptation classes description and their control. The response contains a list of adaptation classes that were matched.

Pre-conditions:

MMSC needs to know if minor and/or major adaptations were performed.

The MMS message contains a SMIL presentation along with a video encoded in video/h263-2000 format encapsulated in video/3gpp file format.

The receiving phone only supports image/jpeg.

STI Request:

…

```
<adaptationClasses>
        <adaptationClass>
            <className>major</className>
            <allowed>true</allowed>
            <classRef>file://Example1-file</classRef>
        </adaptationClass>
        <adaptationClass>
            <className>minor-or-none</className>
            <allowed>true</allowed>
            <classRef>file://Example1-file</classRef>
        </adaptationClass>
</adaptationClasses>
```

…

STI Response:

…

```
<adaptationsPerformed>
        <adaptationPerformed>major</adaptationPerformed>
<adaptationsPerformed>
```

…

Note: the Application Platform would distinguish between an MMS minor adaptation and no adaptation at all by considering warning codes returned (basically a warning telling that no adaptation was performed, see section 5.3.4).

E.2 Web Browsing Adaptation Classes

Another example would be for a Web Portal which wants to learn if layout conversion was performed to convert HTML to any other flavour of HTML, xHTML or any text; or if any video was converted to an image; or if images were deleted:

STI Request:

…

```
<adaptationClasses>
        <adaptationClass>
            <className> LayoutAdaptation </className>
            <allowed>true</allowed>
            <classRef>Example2-file</classRef>
        </adaptationClass>
        <adaptationClass>
            <className> VideoToImage </className>
            <allowed>true</allowed>
            <classRef>Example2-file</classRef>
        </adaptationClass>
</adaptationClasses>
```

STI Response (given that the content adapted met the first two conditions):

…

```
<adaptationsPerformed>
        <adaptationPerformed>LayoutAdaptation</adaptationPerformed>
        <adaptationPerformed>VideoToImage</adaptationPerformed>
</adaptationsPerformed>
```

…

Appendix F. Request Examples (Informative)

F.1 Multipart Request

POST /MMSTrRequest HTTP/1.1

Host: mmsc.host.com:80

Content-Type: multipart/related; boundary=<boundary_string>; type=text/xml; start=”<soap-part>”

SOAPAction: “”

--<boundary_string>

Content-type: text/xml; charset=”utf-8”

Content-Length: ###

Content-Id: <soap-part>

Content-Transfer-Encoding: 8bit

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns="http://www.openmobilealliance.org/schema/sti/v1_0"
xmlns:env=http://schemas.xmlsoap.org/soap/envelope/>
    <env:Body>
        <TranscodingRequest>
            <originatorID>mmsc11</originatorID>
            <operationID>465156321</operationID>
            <profileID>http://nds.nokia.com/uaprof/N7650r100.xml</profileID>
            <applicationType>MMS</applicationType>
            <adaptationClasses>
                <adaptationClass>
                    <className>mms_minor</className>
                    <allowed>true</allowed>
                </adaptationClass>
                <adaptationClass>
                    <className>mms_major</className>
                    <allowed>true</allowed>
                </adaptationClass>
            <adaptationClasses>
            <policyRef>http://100.100.100.100/policy1</policyRef>
                <transcodingJob>
                    <jobID>job1</jobID>
                    <source>
                        <contentType>application/vnd.wap.multipart.related</contentType>
```

```
                    <contentTypeParams>
                        <property>
                            <name>start</name>
                            <value>123456</value>
                        </property>
                        <property>
                            <name>type</name>
                            <value>application/smil</value>
                        </property>
                    </contentTypeParams>
                    <location>FILE://disks/disk0/input/msg184</location>
                </source>
                <target>
                    <externalLocation>
                        <path>FILE://disks/disk0/output/</path>
                        <name> msg184_7650</name>
                    </externalLocation>
                    <transcodingParams>
                        <multipart>
                            <contentType>multipart/related</contentType>
                            <sizeLimit>20000</sizeLimit>
                        </multipart>
                    </transcodingParams>
                </target>
            </transcodingJob>
        </TranscodingRequest>
    </env:Body>
</env:Envelope>
--<boundary_string>--
```

F.2 Request for image, audio and text transcoding using different options

The following example contains a Transcoding Request with three Transcoding Jobs – the image and text jobs have explicit transcoding parameters, and the job with the audio media uses a predefined profile.

```
POST /MMSTrRequest HTTP/1.1
Host: mmsc.host.com:80
Content-Type: multipart/related; boundary=<boundary_string>; type=text/xml; start="<soap-part>"
SOAPAction: ""
```

```
--<boundary_string>
Content-type: text/xml; charset=”utf-8”
Content-Length: ###
Content-Id: <soap-part>
Content-Transfer-Encoding: 8bit

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns="http://www.openmobilealliance.org/schema/sti/v1_0"
xmlns:env=http://schemas.xmlsoap.org/soap/envelope/>
    <env:Body>
        <TranscodingRequest>
            <originatorID>downloadApp1</originatorID>
            <operationID>654164615</operationID>
            <transcodingJob>
                <jobID>job1</jobID>
                <source>
                    <contentType>image/gif</contentType>
                    <location>cid:image1.gif</location>
                </source>
                <target>
                    <transcodingParams>
                        <image>
                            <contentType>image/jpeg</contentType>
                            <sizeLimit>30000</sizeLimit>
                            <transformations>
                                <transformation>
                                    <type> NoiseReduction </type>
                                </transformation>
                                < !-- First rotate the image and then mirror it ->
                                <transformation>
                                    <type> Rotation </type>
                                    <attributes>
                                        <property>
                                            <name>clockwiseAngle</name>
                                            <value>90</value>
                                        </property>
                                    </attributes>
                                    <order>1</order>
```

```
                    </transformation>
                    <transformation>
                        <type> Mirror </type>
                        <attributes>
                            <property>
                                <name>axis</name>
                                <value>LR</value>
                            </property>
                        </attributes>
                        <order>2</order>
                    </transformation>
                </transformations>
                <colorScheme>
                    <scheme>True</scheme>
                    <depth>24</depth>
                </colorScheme>
                <width>101</width>
                <height>80</height>
                <resizeDirective>AspectRatio</resizeDirective>
            </image>
        </transcodingParams>
    </target>
</transcodingJob>
<transcodingJob>
    <jobID>job2</jobID>
    <source>
        <contentType>audio/wav</contentType>
        <location>FILE://disks/disk0/input/aud252.wav</location>
    </source>
    <target>
        <profileID>T68i_goodquality</profileID>
    </target>
</transcodingJob>
<transcodingJob>
    <jobID>job3</jobID>
    <source>
        <contentType>text/plain</contentType>
        <contentTypeParams>
            <property>
```

```
                        <name>charset</name>
                        <value>utf-16</value>
                    </property>
                </contentTypeParams>
                <location>FILE://disks/disk0/input/hello_friend.txt</location>
            </source>
            <target>
                <transcodingParams>
                    <text>
                        <contentType>text/plain</contentType>
                        <contentTypeParams>
                            <property>
                                <name>charset</name>
                                <value>iso-8859-8</value>
                            </property>
                        </contentTypeParams>
                    </text>
                </transcodingParams>
            </target>
        </transcodingJob>
    </TranscodingRequest>
  </env:Body>
</env:Envelope>

--<boundary_string>
Content-type: image/gif;
Content-Length: 15435
Content-Id: image1.gif
Content-Transfer-Encoding: 8bit

                    Image comes here


--<boundary_string>--
```

F.3 Request for transcoding of a video to a multipart

The following example contains a Transcoding Request with one Transcoding Job. A video is transcoded into a multipart. The multipart will contain an image sequence (the requested number of images is 5), an audio part, and a SMIL presentation.

```
POST /MMSTrRequest HTTP/1.1
Host: mmsc.host.com:80
Content-Type: multipart/related; boundary=<boundary_string>; type=text/xml; start=”<soap-part>”
SOAPAction: ""

--<boundary_string>
Content-type: text/xml; charset=”utf-8”
Content-Length: ###
Content-Id: <soap-part>
Content-Transfer-Encoding: 8bit

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns="http://www.openmobilealliance.org/schema/sti/v1_0"
xmlns:env=http://schemas.xmlsoap.org/soap/envelope/>
    <env:Body>
        <transcodingRequest>
            <originatorID>App15673</originatorID>
            <operationID>654164678</operationID>
            <transcodingJob>
                <jobID>theJobId</jobID>
                <source>
                    <contentType>video/3gpp</contentType>
                    <location>cid:video1.3gp</location>
                </source>
                <target>
                    <transcodingParams>
                        <profileID>T68i_goodquality</profileID>
                        <sizeLimit>30000</sizeLimit>
                            <multipart>
                                <contentType>multipart/related</contentType>
                                <transformations>
                                    <transformation>
                                        <type> NumberOfFrames </type>
                                        <attributes>
                                            <property>
                                                <name>totalFrames</name>
                                                <value>5</value>
                                            </property>
```

```
                                        <attributes>
                                    </transformation>
                                </transformations>
                                <presentation>
                                  <contentType>application/smil</contentType>
                                </presentation>
                                <audio>
                                  <contentType>audio/amr</contentType>
                                  <codec>audio/amr</codec>
                                  <bitRate>12200</bitRate>
                                  <samplingRate>8000</samplingRate>
                                </audio>
                                <image>
                                  <contentType>image/jpeg</contentType>
                                  <width>160>/width>
                                  <height>120</height>
                                  <resizeDirective>Crop</resizeDirective>
                                </image>
                            </multipart>
                        </transcodingParams>
                    </target>
                </transcodingJob>
            </TranscodingRequest>
    </env:Body>
</env:Envelope>

--<boundary_string>
Content-type: video/3gpp;
Content-Length: 40587
Content-Id: video1.3gp
Content-Transfer-Encoding: 8bit
```

Video comes here

```
--<boundary_string>--
```

Appendix G. Response Examples (Informative)

G.1 Response with image and audio

This is an example of a Transcoding Response with an image, text and audio in it. This is the response for the request from sectionF.2.

Note that since the Audio content element could not fit into the requested size limit, the Transcoding Platform compressed it by truncating its duration.

Also note that although the 3rd job failed the transcoding operation as a whole is considered successful.

HTTP/1.1 200 OK

Content-Type: multipart/related; boundary=”<boundary_string>"; type=text/xml; start=”<soap-part>”

--<boundary_string>

Content-type: text/xml; charset=”utf-8”

Content-Length: ###

Content-Id: <soap-part>

Content-Transfer-Encoding: 8bit

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns="http://www.openmobilealliance.org/schema/sti/v1_0"
xmlns:env=http://schemas.xmlsoap.org/soap/envelope/>
    <env:Body>
        <TranscodingResponse>
            <originatorID>downloadApp1</originatorID>
            <operationID>654164615</operationID>
            <mainReturnResult>
                <returnCode>2000</returnCode>
                <returnString> Success – Successful Result </returnString>
            </mainReturnResult>
            <totalDuration>647</totalDuration>
            <jobResult>
                <jobID>job1</jobID>
                <mainReturnResult>
                    <returnCode>2000</returnCode>
                    <returnString> Success – Successful Result </returnString>
                </mainReturnResult>
                <duration>78</duration>
                <output>
                    <contentType>image/jpeg</contentType>
```

```
                        <location>cid:image1res.jpg</location>
                        <mediaSize>15478</mediaSize>
                    </output>
                </jobResult>
                <jobResult>
                    <jobID>job2</jobID>
                    <mainReturnResult>
                        <returnCode>2006</returnCode>
                        <returnString> Warning – Content truncated to meet sizeLimit </returnString>
                    </mainReturnResult>
                    <duration>509</duration>
                    <output>
                        <contentType>audio/amr</contentType>
                        <location>cid:audio1res.amr</location>
                        <mediaSize>31548</mediaSize>
                    </output>
                </ jobResult>
                <jobResult>
                    <jobID>job3</jobID>
                    <mainReturnResult>
                        <returnCode>5000</returnCode>
                        <returnString>Server Error – Unsupported parameter value: iso-8859-8 </returnString>
                    </mainReturnResult>
                    <duration>5</duration>
                </jobResult>
            </TranscodingResponse>
        </env:Body>
    </env:Envelope>

    --<boundary_string>
    Content-type: image/jpeg;
    Content-Length: 15478
    Content-Id: image1res.jpg
    Content-Transfer-Encoding: 8bit

                        Image comes here

    --<boundary_string>
```

Content-type: audio/amr

Content-Length: 31548

Content-Id: audio1res.amr

Content-Transfer-Encoding: 8bit

Audio comes here

--<boundary_string>--

G.2 Response for failures

HTTP/1.1 500 Internal Server Error

Content-type: text/xml; charset=”utf-8”

Content-Transfer-Encoding: 8bit

Content-Length: ###

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env=http://schemas.xmlsoap.org/soap/envelope/>
    <env:Body>
      <soap:Fault xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
          <faultcode>soap:Client</faultcode>
          <faultstring>Any error message.</faultstring>
          <faultactor>http://www.site.com/sti/v1_0</faultactor>
          <detail>
              <TranscodingResponse xmlns="http://www.openmobilealliance.org/schema/sti/v1_0" >
                  <originatorID>app1</originatorID>
                  <operationID>544868093</operationID >
                  <mainReturnResult>
                      <returnCode>4001</returnCode>
                      <returnString> Client Error – Parsing error: Invalid STI Request </returnString>
                  </mainReturnResult>
                  <totalDuration>35</totalDuration>
              </TranscodingResponse>
          </detail>
      </soap:Fault>
    </env:Body>
</env:Envelope>
```

Appendix H. Examples of transcodingParams and transformations Combinations (Informative)

The following table shows how various combinations of structures and parameters in *transcodingParams* and *transformations* can be used in order to obtain specific output.

No.	Input	Desired Output	transcodingParams Media Element	Usage of transformations
1.	Video (standalone or in multipart)	Video (standalone)	Video	
2.	Video (standalone or in multipart)	Video in multipart	Video within a Multipart	
3.	Video or animation (standalone or in multipart)	Single Image (any format other than GIF)	Image	Offset=n (Optional) - To specify which image to choose as single image. - If no offset, the image is chosen by the Transcoding Platform (implementation specific).
4.	Video or animation (standalone or in multipart)	Single Image in GIF format	Image - Specify one of the GIF content types mode	(Same as use case 3) + (Image)NumberOfFrames=1 - If (Image)NumberOfFrames is not specified, the result could be an animated GIF. If it is specified and is greater than one, the result will be an animation with the given number of frames.
5.	Video or animation (standalone or in multipart)	Multiple Images in a Multipart (any format other than GIF)	Image within a Multipart - (optional) Specify a presentation element to create a slide-show of images, otherwise only a collection of images is created (no SMIL).	(Multipart)NumberOfFrames=n - If number of frames is 1, only one image as output (use case 7). - If number of frames is 0 or not specified, the number of images in the output is chosen by the Transcoding Platform (implementation-specific).
6.	Video or animation (standalone or in multipart)	Multiple Images in GIF format in a Multipart	Image within a Multipart - Specify one of the GIF content types - (optional) Specify a presentation element to create a slide-show of images, otherwise only a collection of images is created (no SMIL).	(Multipart)NumberOfFrames=n - If number of frames is 1, only one slide as output (use case 8), but that slide may be a single GIF image or a GIF animation depending on the (Image)NumberOfFrames parameter. - If number of frames is 0 or not specified, the number of images in the output is chosen by the Transcoding Platform (implementation-specific). (Image)NumberOfFrames=1 - If (Image)NumberOfFrames is not specified, the result could be several animated GIFs. If it is specified and is

				greater than one, the result will be several animated GIFs, each with the given number of frames.
7.	Video or animation (standalone or in multipart)	Single Image in a Multipart (any format other than GIF)	Image within a Multipart - (optional) Specify a presentation element to put the image in a presentation, otherwise only one image alone is created (no SMIL).	(Multipart)NumberOfFrames=1
8.	Video or animation (standalone or in multipart)	Single Image in GIF format in a Multipart	Image within a Multipart - Specify one of the GIF content types - (optional) Specify a presentation element to put the image in a presentation, otherwise only one image alone is created (no SMIL).	(Multipart)NumberOfFrames=1 (Image)NumberOfFrames=1 - If (Image)NumberOfFrames is not specified, the result could be an animated GIF. If it is specified and is greater than one, the result will be an animated GIF with the given number of frames (see use case 10).
9.	Video (standalone or in multipart)	Animation (GIF)	Image - Specify animated GIF content type	(Image)NumberOfFrames=n (Image)FrameRateOutput - both are optional. Even if (Image)NumberOfFrames is not specified, the result may still be an animation.
10.	Video (standalone or in multipart)	(one) Animation (GIF) in a Multipart	Image within a Multipart - Specify animated GIF content type. - (optional) Specify a presentation element to put the image in a presentation, otherwise only one image alone is created (no SMIL).	(Multipart)NumberOfFrames=1 - If number of frames is 0 or not specified, the number of animated gif in the output is chosen by the Transcoding Platform (implementation specific). (Image)NumberOfFrames=n (Image)FrameRateOutput
11.	Video (standalone or in multipart)	Several Animations (GIF) in a Multipart	Image within a Multipart - Specify animated GIF content type. - (optional) Specify a presentation element to put the image in a presentation, otherwise only one image alone is created (no SMIL).	(Multipart)NumberOfFrames=n - If number of frames is 0 or not specified, the number of animated gif in the output is chosen by the Transcoding Platform (implementation specific). (Image)NumberOfFrames=n Image)FrameRateOutput
12.	Video (standalone or in multipart)	Audio only	Audio e.g. 1) audio/3gpp for audio in .3gp 2) audio/amr for audio in .amr	
13.	Video (standalone or in multipart)	Audio only in a Multipart	Audio within a Multipart Video with sizeLimit=0 within a Multipart	

14.	Video (standalone or in multipart)	Audio only in video in a Multipart	Video within a Multipart VideoVisual with sizeLimit=0 VideoAudio	
15.	Video (standalone or in multipart)	Audio only in a Video	Video VideoVisual with sizeLimit=0 VideoAudio	
16.	Video (standalone or in multipart)	Visual part only	Video VideoVisual VideoAudio with sizeLimit=0	
17.	Video (standalone or in multipart)	Visual part only in a Multipart	Video within a Multipart VideoAudio with sizeLimit=0	

Appendix I. Usage of min-bit-rate, encoding-method, and *bitRate*

The following table shows the relationship between the encoding-method codec parameter, the min-bit-rate codec parameter and the *bitRate* parameter.

INPUT			RESULT		
encoding-method	**min-bit-bate**	**bitRate**	**encoding-method**	**Range or Value**	**Note**
Not specified	Not specified	Specified	FBR (or CBR)	bitRate	
Not specified	Specified	Specified	VBR	min-bit-rate to bitRate	
Not specified	Not specified	Not specified	Default	Default	1
Not specified	Specified	Not specified	VBR	min-bit-rate to full	2
VBR	Not specified	Specified	VBR	lowest bit rate to bitRate	3
VBR	Not specified	Not specified	VBR	lowest to full	2,3
VBR	Specified	Not specified	VBR	min-bit-rate to full	2
VBR	Specified	Specified	VBR	min-bit-rate to bitRate	
FBR	Not specified	Specified	FBR	bitRate	
FBR	Not specified	Not specified	FBR	Default	1
FBR	Specified	Not specified	FBR	Default	1,4
FBR	Specified	Specified	FBR	bitRate	4

Comments:

1) Implementation-specific
2) full or implementation-specific
3) lowest or implementation-specific
4) min-bit-rate ignored

Appendix J. Alphabetical Index of Parameters Tables

The tables in this appendix serve as an alphabetical index to the STI parameters. Details about each parameter and its usage can be found in the tables within the spec.

J.1 Transcoding Request Alphabetical Index (*Table 1*)

No.	Parent	Name	Type	Possible Values	Mandatory/ Optional
1.	adaptationClass	allowed	Boolean		Optional (0..1)
2.	adaptationClass	className	Token		Mandatory (1..1)
3.	adaptationClass	classRef	URI		Optional (0..1)
4.	adaptationClasses	**adaptationClass**			Optional (0..n)
5.	externalLocation	name	Token		Optional (0..1)
6.	externalLocation	path	anyURI		Mandatory (1..1)
7.	source	contentType	Token		Mandatory (1..1)
8.	source	**contentTypeParams** (see ***Table 7***)			Optional (0..1)
9.	source	**extensionData** (see ***Table 7***)			Optional (0..1)
10.	source	location	anyURI	URI	Mandatory (1..1)
11.	target	applicationSizeLimit	unboundedLong	In bytes	Optional (0..1)
12.	target	applicationType	String		Optional (0..1)
13.	target	**extensionData** (see ***Table 7***)			Optional (0..1)
14.	target	**externalLocation**			Optional (0..1)
15.	target	profileID	String		Optional (0..1)
16.	target	**transcodingParams** (see *Table 7*)			Optional (0..1)
17.	transcodingJob	**adaptationClasses**			Optional (0..1)
18.	transcodingJob	**extensionData** (see ***Table 7***)			Optional (0..1)
19.	transcodingJob	jobID	Token		Mandatory (1..1)
20.	transcodingJob	policyRef	anyURI	URI	Optional (0..1)

21.	transcodingJob	**source**			Mandatory (1..1)
22.	transcodingJob	**target**			Optional (0..1)
23.		**TranscodingRequest**			Mandatory (1..1)
24.	TranscodingRequest	**adaptationClasses**			Optional (0..1)
25.	TranscodingRequest	applicationSizeLimit	unboundedLong	In bytes	Optional (0..1)
26.	TranscodingRequest	applicationType	String		Optional (0..1)
27.	TranscodingRequest	**extensionData** (see ***Table 7***)			Optional (0..1)
28.	TranscodingRequest	operationID	Token		Mandatory (1..1)
29.	TranscodingRequest	originatorID	Token		Mandatory (1..1)
30.	TranscodingRequest	policyRef	anyURI	URI	Optional (0..1)
31.	TranscodingRequest	profileID	String		Optional (0..1)
32.	TranscodingRequest	**transcodingJob**			Mandatory (1..n)
33.	TranscodingRequest	**transcodingParams** (see ***Table 7***)			Optional (0..1)

J.2 TranscodingParams Alphabetical Index (*Table 7*)

No.	Parent	Name	Type	Units/ Possible Values	Mandatory/ Optional
1.		**transcodingParams**			Optional (0..1)
2.	attributes	**property**			Optional (0..n)
3.	audio	bitRate	nonNegativeInt	In bps	Optional (0..1)
4.	audio	channels	Token	“Mono”, “Stereo”, “DualMono”, “IntensityStereo”[11]	Optional (0..1)
5.	audio	codec	Token		Optional (0..1)
6.	audio	**codecParams**			Optional (0..1)
7.	audio	contentType	Token		Optional

[11] This list is not exhaustive and other values can be used in order to express additional and/or proprietary information.

					(0..1)
8.	audio	**contentTypeParams**			Optional (0..1)
9.	audio	**extensionData**			Optional (0..1)
10.	audio	samplingRate	nonNegativeInt	In Hz	Optional (0..1)
11.	audio	samplingResolution	nonNegativeInt	In bits per sample	Optional (0..1)
12.	audio	sizeLimit	unboundedLong	In bytes	Optional (0..1)
13.	audio	**synthetic**			Optional (0..1)
14.	audio	**transformations**			Optional (0..1)
15.	codecParams	**property**			Optional (0..n)
16.	colorScheme	depth	nonNegativeInt		Mandatory (1..1)
17.	colorScheme	scheme	Token	"True", "PaletteColor" "PaletteGrey"[12]	Mandatory (1..1)
18.	contentTypeParams	**property**			Optional (0..n)
19.	extensionData	**property**			Optional (0..n)
20.	image	codec	Token		Optional (0..1)
21.	image	**codecParams**			Optional (0..1)
22.	image	**colorScheme**			Optional (0..1)
23.	image	contentType	Token		Optional (0..1)
24.	image	**contentTypeParams**			Optional (0..1)
25.	image	**extensionData**			Optional (0..1)
26.	image	height	nonNegativeInt	In pixels	Optional (0..1)
27.	image	resizeDirective	Token	"AspectRatio", "Crop", "Stretch"[13]	Optional (0..1)

[12] This list is not exhaustive and other values can be used in order to express additional and/or proprietary information.

[13] This list is not exhaustive and other values can be used in order to express additional and/or proprietary information.

28.	image	sizeLimit	unboundedLong	In bytes	Optional (0..1)
29.	image	**transformations**			Optional (0..1)
30.	image	upsizeAllowed	Boolean		Optional (0..1)
31.	image	width	nonNegativeInt	In pixels	Optional (0..1)
32.	multipart	**audio**			Optional (0..1)
33.	multipart	contentType	Token		Optional (0..1)
34.	multipart	**contentTypeParams**			Optional (0..1)
35.	multipart	**extensionData**			Optional (0..1)
36.	multipart	**image**			Optional (0..1)
37.	multipart	**presentation**			Optional (0..1)
38.	multipart	sizeLimit	unboundedLong	In bytes	Optional (0..1)
39.	multipart	**text**			Optional (0..1)
40.	multipart	**transformations**			Optional (0..1)
41.	multipart	**video**			Optional (0..1)
42.	presentation	contentType	Token		Optional (0..1)
43.	presentation	**contentTypeParams**			Optional (0..1)
44.	presentation	layout	Token	“Portrait”, “Landscape”	Optional (0..1)
45.	presentation	template	anyURI		Optional (0..1)
46.	property	name	Token		Mandatory (1..1)
47.	property	value	String		Optional (0..1)
48.	synthetic	channelsPriority	Token		Optional (0..1)
49.	synthetic	channelToUse	nonNegativeInt		Optional (0..1)
50.	synthetic	instrument	nonNegativeInt		Optional (0..1)

51.	text	contentType	Token		Optional (0..1)
52.	text	**contentTypeParams**			Optional (0..1)
53.	text	**extensionData**			Optional (0..1)
54.	text	sizeLimit	unboundLong	In bytes	Optional (0..1)
55.	text	**transformations**			Optional (0..1)
56.	transcodingParams	**audio**			Optional (0..1)
57.	transcodingParams	**extensionData**			Optional (0..1)
58.	transcodingParams	**image**			Optional (0..1)
59.	transcodingParams	**multipart**			Optional (0..1)
60.	transcodingParams	sizeLimit	unboundedLong	In bytes	Optional (0..1)
61.	transcodingParams	**text**			Optional (0..1)
62.	transcodingParams	**video**			Optional (0..1)
63.	transformation	**attributes**			Optional (0..1)
64.	transformation	order	nonNegativeInt		Optional (0..1)
65.	transformation	type	Token		Mandatory (1..1)
66.	transformations	**transformation**			Optional (0..n)
67.	video	contentType	Token		Optional (0..1)
68.	video	**contentTypeParams**			Optional (0..1)
69.	video	**extensionData**			Optional (0..1)
70.	video	sizeLimit	unboundedLong	In bytes	Optional (0..1)
71.	video	**transformations**			Optional (0..1)
72.	video	**videoAudio**			Optional (0..1)
73.	video	**videoVisual**			Optional (0..1)
74.	videoAudio	bitRate	nonNe	In bps	Optional

			gativeInt		(0..1)
75.	videoAudio	channels	Token	"Mono", "Stereo", "DualMono", "IntensityStereo"[14]	Optional (0..1)
76.	videoAudio	codec	Token		Optional (0..1)
77.	videoAudio	**codecParams**			Optional (0..1)
78.	videoAudio	samplingRate	nonNegativeInt	In Hz	Optional (0..1)
79.	videoAudio	samplingResolution	nonNegativeInt	In bits per sample	Optional (0..1)
80.	videoAudio	sizeLimit	unboundedLong	In bytes	Optional (0..1)
81.	videoAudio	**transformations**			Optional (0..1)
82.	videoVisual	bitRate	nonNegativeInt	In bps	Optional (0..1)
83.	videoVisual	codec	Token		Optional (0..1)
84.	videoVisual	**codecParams**			Optional (0..1)
85.	videoVisual	frameRate	Float	In frames per second	Optional (0..1)
86.	videoVisual	height	nonNegativeInt	In pixels	Optional (0..1)
87.	videoVisual	resizeDirective	Token	"AspectRatio", "Crop", "Stretch"[15]	Optional (0..1)
88.	videoVisual	sizeLimit	unboundedLong	In bytes	Optional (0..1)
89.	videoVisual	**transformations**			Optional (0..1)
90.	videoVisual	upsizeAllowed	Boolean		Optional (0..1)
91.	videoVisual	width	nonNegativeInt	In pixels	Optional (0..1)

J.3 Transcoding Response Alphabetical Index (*Table 8*)

No.	Parent	Name	Type	Units /	Mandatory / Optional

[14] This list is not exhaustive and other values can be used in order to express additional and/or proprietary information.

[15] This list is not exhaustive and other values can be used in order to express additional and/or proprietary information.

				Possible Values	
1.	**adaptationsPerformed**	adaptationPerformed	Token		Optional (0..n)
2.	**additionalReturnResults**	**returnResult**			Optional (0..n)
3.	jobResult	**adaptationsPerformed**			Optional (0..1)
4.	jobResult	**additionalReturnResults**			Optional (0..1)
5.	jobResult	duration	nonNegativeInt	In milliseconds	Optional (0..1)
6.	jobResult	**extensionData** (see ***Table 7***)			Optional (0..1)
7.	jobResult	jobID	Token		Mandatory (1..1)
8.	jobResult	**mainReturnResult**			Mandatory (1..1)
9.	jobResult	**output**			Conditional (0..1)
10.	mainReturnResult	returnCode	Integer		Mandatory (1..1)
11.	mainReturnResult	returnString	String		Optional (0..1)
12.	output	contentType	Token		Mandatory (1..1)
13.	output	**contentTypeParams** (see Table 7)			Optional (0..1)
14.	output	**extensionData** (see ***Table 7***)			Optional (0..1)
15.	output	location	anyURI	URI	Mandatory (1..1)
16.	output	mediaSize	nonNegativeLong	In bytes.	Optional (0..1)
17.	returnResult	returnCode	Integer		Mandatory (1..1)
18.	returnResult	returnString	String		Optional (0..1)
19.		**TranscodingResponse**			Conditional (0..1)
20.	TranscodingResponse	**additionalReturnResults**			Optional (0..1)

21.	TranscodingResponse	**extensionData** (see *Table 7*)			Optional (0..1)
22.	TranscodingResponse	**jobResult**			Conditional (0.. n)
23.	TranscodingResponse	**mainReturnResult**			Mandatory (1..1)
24.	TranscodingResponse	operationID	Token		Conditional (0..1)
25.	TranscodingResponse	originatorID	Token		Conditional (0..1)
26.	TranscodingResponse	totalDuration	nonNegativeInt	In milliseconds	Mandatory (1..1)

J.4 Transformations Alphabetical Index (*Table 6*)

No.	Transformation Type	Media Element	Attribute Name	Possible attribute values	Mandatory / Optional attribute	Value Constraints
1.	AGC	Audio, VideoAudio	No attribute			
2.	Brightness	Image, VideoVisual	level	integer in the range -50..50[16] (inclusive)	M	Desired
3.	Contrast	Image, VideoVisual	level	integer in the range -50..50[3] (inclusive)	M	Desired
4.	Color	Image, VideoVisual	level	integer in the range -50..50[3] (inclusive)	M	Desired
5.	Cropping	Image, VideoVisual				
			top	a non negative integer value	M	Exact
			left	a non negative integer value	M	Exact
			bottom	a non negative integer value	M	Exact
			right	a non negative integer value	M	Exact
6.	DurationLimit	Video, Audio, Multipart	limit	a positive integer value	M	Maximum
7.	FrameFill	Image,				

[16] The scale (-50..50) is a relative scale of intensity or level of the correction, and it is not defined by specific physical measure.

		VideoVisual	R	integer in range 0..255	M	Exact
			G	integer in range 0..255	M	Exact
			B	integer in range 0..255	M	Exact
8.	FrameRateOutput	VideoVisual, Multipart	fps	a positive float value		Exact
9.	FrameRateSample	Image, VideoVisual, Multipart	fps	a positive float value		Exact
10.	LevelCorrection	Image, VideoVisual	No attribute			
11.	Mirror	Image, VideoVisual	axis	"UD", "LR"	M	Exact
12.	NoiseReduction	Image, VideoVisual	No attribute			
13.	NumberOfFrames	Image, VideoVisual, Multipart	totalFrames	a positive integer value		Exact
14.	Offset	Image, Audio, Video	startTime	a positive integer value		
15.	Rotation	Image, VideoVisual	clockwiseAngle	90 180 270 auto	M	Exact (Except for the 'auto' value)
16.	Sharpen	Image, VideoVisual	No attribute			