SMPTE STANDARD

Media Dispatch Protocol (MDP) — Basic Target Pull Profile Specification

Table of Contents

Approved
December 14, 2007

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in Part XIII of its Administrative Practices.

SMPTE Standard 2032-3 was prepared by Technology Committee S22 on Committee on Television Systems Technology.

Introduction

This section is entirely informative and does not form an integral part of this document.

The Media Dispatch Protocol (MDP) is a means of orchestrating the delivery of media files over IP networks. It defines a standard mechanism for implementations to initiate a delivery, to negotiate the details of the delivery, to provide information about the progress of the delivery, and to provide a confirmation of the outcome of the delivery.

MDP allows organizations to transfer files at agreed times, using agreed transfer protocols, and using an agreed set of secure technologies (where appropriate). However, the protocol is not overly prescriptive in regards to which protocols or technologies shall be used; rather it provides a framework which allows organizations to choose those that best suit their own needs.

MDP is a multi-part standard:

Part 1, the MDP protocol specification, defines the logical messages, data structures and data dictionary of MDP.

Part 2, an MDP mapping specification, defines a representation of the messages and structures defined in Part 1.

Part 3, of which this document is part, defines subsets of the protocol and rules on the use of these subsets.

Part 4, the MDP Engineering Guideline, introduces and describes MDP and how it can be used in particular scenarios.

1 Scope

This standard defines the MDP Basic Target Pull Profile. This specifies a subset of the MDP protocol defined in SMPTE 2032-1, in which the name and size of all files is known to the initiating agent before the transaction is initiated, the target agent initiates the file transfers, and all transfers use a pull-mode protocol.

The standard specifies the sequence of messages that shall or may be used in a transaction, how certain properties shall be set in the manifest document, how the file transfers shall be initiated, and which transfer protocols an agent shall support.

2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword “reserved” indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword “forbidden” indicates “reserved” and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

3 Normative Reference

The following standard contains provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the edition indicated was valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standard indicated below.

SMPTE 2032-1-2007, Media Dispatch Protocol (MDP) — Protocol Specification

4 Acronyms and Definitions

The full glossary of acronyms and definitions used in the MDP specification is given in the MDP protocol specification (SMPTE 2032-1). It is not repeated here to avoid any divergence of meaning.

5 Background (Informative)

The MDP protocol standard (SMPTE 2032-1) defines the complete set of messages and data structures available to MDP agents. An MDP profile standard defines a subset of these messages and structures, and specifies rules for their use during a transaction. It allows the complexity of an agent implementation to be appropriate to the delivery scenario.

Two agents that correctly implement the same profile, and also the same MDP mapping standard, will be able to interoperate correctly. A pair of agents can determine, via the capabilities document defined in the protocol whether they implement a compatible profile, prior to initiating a transaction.

This profile provides a suitable subset of features for the common "target pull" scenario. In this scenario, the initiator agent has available all relevant information about the files to be transferred, the target agent is responsible for initiating all file transfers, and all transfers use a pull-mode protocol

6 Summary of Profile (Informative)

A transaction compliant with this profile has the following characteristics:

- The name and size of all files is known to the initiating agent before the transaction is initiated.
- The target agent initiates all file transfers.
- All transfers use a pull-mode protocol.
- A simple negotiation procedure is used in which the initiator agent sends a manifest document containing all the `file` and `transferoption` and `transferwith` elements. The target agent then sends a manifest document accepting or rejecting each `transferwith` element.
- The target agent sends a manifest document to notify that the first transfer has started.
- An agent may request a capabilities document from the other agent, which replies synchronously.
- The initiator agent may request a manifest document from the target agent, which replies synchronously.
- The initiator agent may request pause and resume of transfers.

Annex A of the MDP protocol specification (SMPTE 2032-1) includes a figure showing an example of the messages sent in a transaction compliant with this profile.

Agents supporting this profile are able to initiate file transfers using HTTP GET and HTTP/TLS GET.

7 Properties in Manifest Documents

The `profile` property of the `manifest` element shall have the value “`basic target pull`”.

The `usecase` property of the `manifest` element shall have the value “`target pull`”.

The `direction` property of all `transferoption` elements shall have the value `PULL`.

The `sender` property shall be present in all `transferoption` elements.

The receiver property shall not be present.

The controller property of all transferoption elements shall have the same value as the target property of the manifest element.

The pathname, size and mimetype properties shall be present in all file elements.

8 Use of Protocol in a Transaction

8.1 Capabilities

An agent may send a qry_requestcapabilities query to another agent at any time.

Note: Typically this will be done before the initiation of a transaction.

The agent receiving the query shall respond with an rpl_capabilities or rpl_error reply. The agent should provide as complete a list of capabilities as possible in an rpl_capabilities reply.

8.2 Initiation and Negotiation

It is assumed that the target agent's message endpoint is already known to the initiator agent.

The following sequence of steps shall be used for initiation of a transaction and negotiation of delivery details:

1. The initiator agent shall send a cmd_initiatingtransaction command to the target agent's message endpoint. The manifest parameter of this command shall contain all proposed file, transferoption and transferwith elements for this transaction. The agent parameter shall specify the initiator agent's message endpoint for the transaction.

2. The target agent shall either agree the initiation of the transaction with a cnf_ok, or shall send a cnf_error.

3. The target agent shall send a cmd_sendingmanifest command to the initiator agent's message endpoint. The manifest parameter shall comply with each of the following:

    a) Each transferwith element shall have a status property with value accepted or rejected.

    b) At most one transferwith element shall have its status property set to accepted for a given file.

    c) No other change shall be made with respect to the manifest document sent in step 1, other than to the status properties, and to the "updated" elements within transferwith elements.

4. The initiator agent shall acknowledge the details in the manifest document sent at step 3 with a cnf_ok, or shall send a cnf_error. In the latter case the initiator agent should terminate the transaction by sending a cmd_abortingtransaction command.

5. If no file element in the manifest document sent at step 3 contains a transferwith element whose status property has value accepted (all proposals were rejected), the target agent shall complete the transaction as in section 8.7.

8.3 Transfer

8.3.1 General

The target agent shall initiate a transfer for each file that has a `transferwith` element whose `status` property has value `accepted`.

Each transfer shall use the protocol specified by the `protocol` property of the relevant `transferoption` element.

8.3.2 Transfer endpoint

The target agent shall perform each file transfer by means of the URL generated by concatenating: the `sender` property of the appropriate `transferoption` element, the "`/`" character (if required syntactically), and the `relativepath` property of the relevant `file` element.

8.3.3 Properties of transferred files

The name of each file transferred shall be the same as the filename part of the `pathname` property of the relevant `file` element. This is the final part of the `pathname`, after any string leading up to the final "`/`" character, if present, has been removed.

The size of each file transferred shall be the same as the `size` property of the relevant `file` element.

8.3.4 Timescale

Each transfer shall start no earlier than the later of the times specified by the `startafter` property of the relevant `file` element, and the `availablefrom` property of the relevant `transferoption` element.

8.3.5 Priorities

The `priority` property values of `file` elements should determine the order in which the corresponding file transfers are initiated within a transaction. The target agent should not initiate a transfer until it has attempted to initiate all transfers with lower priority values. A transfer without a `priority` property should not be initiated until after all transfers with `priority` properties have been initiated.

8.4 Notification of Initiation of Transfer

After initiating at least one transfer, the target agent shall send a `cmd_sendingmanifest` command to the initiator agent's message endpoint. This should be sent as soon as possible after the first transfer has started. The `manifest` parameter shall indicate which transfer(s) have started by means of the `status` property of each `transferwith` element.

The initiator agent shall acknowledge the details in the `manifest` document sent with a `cnf_ok`, or shall send a `cnf_error`. In the latter case, the initiator agent should terminate the transaction by sending a `cmd_abortingtransaction` command.

8.5 Status Check

The initiator agent may send a `qry_requestmanifest` query to the target agent's message endpoint at any time after negotiation is completed and before the transaction is completed or terminated. The query shall not be sent at other times.

The target agent shall reply with an `rpl_manifest`, or an `rpl_error`. In the case that an `rpl_manifest` is sent, the `manifest` document payload shall indicate the status of the transaction when the reply is made: the `status` property of each `transferwith` element shall indicate the current status of the file transfer, and the `bytestransferred` property of each `file` element shall indicate the current number of bytes that have been received for the file.

The target agent shall not send a `qry_requestmanifest` query.

8.6 Pause and Resume

The initiator agent may send a `cmd_pausetransfers` or `cmd_resumetransfers` command to the target agent's message endpoint at any time after negotiation is completed and before the transaction is completed or terminated. The transfers to be paused or resumed shall be identified by the `file_list` parameter. If the `file_list` parameter is omitted then all transfers shall be paused or resumed.

The target agent may pause or resume the transfer or transfers. It shall respond with a `cnf_ok` or a `cnf_error`.

Note: The initiator agent can subsequently check whether the transfers have been paused or resumed as in section 8.5.

8.7 Completion

After all file transfers in the transaction have succeeded or failed, the target agent shall send a `cmd_completingtransaction` command to the initiator agent's message endpoint. This shall include a `manifest` parameter in which: the `bytestransferred` property of each `file` element shall signal the exact number of bytes received for the file, and the `status` property of each `transferwith` element shall signal the result of the transfer. If the number of bytes received does not match the `size` property for the `file` element, then the status shall be `failed`, and the `updatereason` property shall be `INCORRECT_FILE_SIZE`.

The initiator agent shall respond with a `cnf_ok` or a `cnf_error`.

8.8 Termination

Either agent may send a `cmd_abortingtransaction` message at any time after the transaction has been initiated. This shall be sent to the other agent's message endpoint.

The other agent shall respond with a `cnf_ok` or a `cnf_error`.

9 Required Protocols

9.1 HTTP

The target agent shall be able to initiate a file transfer via an HTTP/1.1 GET request. In such cases, the `protocol` property shall take the value of `HTTP`.

Note: This standard does not require agents to use any particular HTTP/1.1 content coding or transfer coding to transfer files. For instance, implementations may make use of "chunked" transfer coding, but this information is not represented within MDP.

At least one HTTP server shall be available with access to the files that the initiator agent lists in the manifest document. The server(s) shall be able to respond to an HTTP/1.1 GET request by transferring the appropriate file data.

Note: The HTTP server can be located on a different host to the initiator agent.

9.2 HTTP/TLS

An agent shall be able to initiate a file transfer via an HTTP/1.1 GET request over TLS. In such cases, the `protocol` property shall be set to `HTTP/TLS`, and the `encrypted` property shall be set to `true`.

Note: This standard does not require agents to adopt any particular mechanism for authenticating each other. Nor does it require agents to use any particular key size, or encryption cipher, or any other aspect of TLS.

At least one HTTP/TLS server shall be available with access to the files that the initiator agent lists in the manifest document. The server(s) shall be able to respond to an HTTP/1.1 GET request over TLS by transferring the appropriate file data.

Note: The HTTP/TLS server can be located on a different host to the initiator agent.

9.3 Other Protocols

Agents may support the sending and receiving of files using other protocols. The protocol property shall take the value defined by the dictionary of the MDP protocol.

Only protocols that support pull-mode transfer shall be used.

Protocols not specified in SMPTE 2032-1 may be used by using a protocol value starting with "`x-`" or "`X-`".

Encryption and authentication mechanisms other than TLS may be used.

10 Integrity Check

10.1 Requesting Integrity Check

The initiator agent may request that the target agent check the integrity of received files.

This shall be done by including in the initial manifest document: an `integritycheck` property with value `true` for each relevant `transferoption` element, a `mustintegritycheck` property with value `true` for each relevant `file` element, and a `hash` element for each relevant `file` element.

10.2 Performing Integrity Check

If requested, the target agent shall perform an integrity check on each file received.

This shall be done by calculating the hash (digest) value of the file using the algorithm specified by the relevant `hashtype` property and comparing the result with the value of relevant `hashvalue` property. If there is a difference, the target agent shall set the `status` property of the relevant `transferwith` element to `failed`, and the `updatereason` property to `INCORRECT_HASH`.

10.3 Required Hash Types

Agents shall be capable of performing an integrity check using the MD5 hash type, and may support other hash types.

Hash types not specified in SMPTE 2032-1 may be used by using a `hashtype` property value starting with "`x-`" or "`X-`".

Note: The transfer protocol might guarantee data integrity. This could be used instead of, or as well as, the integrity check provided by MDP.