SMPTE STANDARD

Media Dispatch Protocol (MDP) — Protocol 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 2032-1 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, this part, defines the logical messages, data structures and data dictionary of MDP.

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

Part 3, the MDP profile specification, 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 message set, data structures and data dictionary of the Media Dispatch Protocol (MDP) for orchestrating the transfer of files across IP networks.

The standard also specifies requirements on mapping specifications that define representations of the messages and data structures.

The standard also defines requirements on profile specifications that define subsets of the protocol and rules on the use of those subsets.

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 References

The following standards contain provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were 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 standards indicated below.

W3C XML Schema 1.1 — Part 2: Datatypes

IETF RFC 1321, The MD5 Message-Digest Algorithm

IETF RFC 2046, Multipurpose Internet Mail Extensions (MIME) — Part Two: Media Types

IETF RFC 2279, UTF-8 — A Transformation Format of ISO 10646

ISO/IEC 3309:1993, Information Technology — Telecommunications and Information Exchange Between Systems — High-Level Data Link Control (HDLC) Procedures — Frame Structure

ISO/IEC 8601:2000, Data Elements and Interchange Formats — Information Interchange — Representation of Dates and Times

ISO/IEC 10646:2003, Information Technology — Universal Multiple-Octet Coded Character Set (UCS)

FIPS 180-2, Secure Hash Signature

4 Acronyms (Informative)

CRC: Cyclic Redundancy Check

FIPS: Federal Information Processing Standards (issued by NIST)

FTP: File Transfer Protocol

HTTP: Hypertext Transfer Protocol

IANA: Internet Assigned Numbers Authority

IETF: Internet Engineering Task Force

ISO: International Organization for Standardization

MD5: Message-Digest Algorithm 5

MDP: Media Dispatch Protocol

MIME: Multipurpose Internet Mail Extensions

MXF: Material Exchange Format

NIST: National Institute of Standards and Technology

RFC: Request For Comment

SFTP: Secure File Transfer Protocol

SHA: Secure Hash Algorithm

TLS: Transport Layer Security

URI: Uniform Resource Identifier

URL: Uniform Resource Locator

UTF-8: 8-bit UCS/Unicode Transformation Format

W3C: World Wide Web Consortium

XML: Extensible Markup Language

5 Definitions

Agent: A software entity that acts on behalf of an organization to implement the Media Dispatch Protocol.

Authenticate: Confirm the identity of an agent.

Command: A message that typically causes change within a transaction or sends information.

Confirmation: A message that provides an immediate positive or negative acknowledgement of a command.

Data structure: A defined piece of information that may be carried within a message. May be an element, property or list.

Dictionary: Allowed values of properties.

Element: A data structure containing other elements, lists or properties. Note: the term “element” is used in other SMPTE documents and might have a different meaning in these.

Encrypt: Secure a message or transfer, for example by means of a cipher.

File: A unit of content, typically a stored media file, though files may be dynamically generated.

Hash type: Algorithm used in an integrity check process.

Initiator agent: The agent that sends a message to initiate a transaction.

Integrity check: Verify that a file has been transferred correctly, for example by means of a hashing algorithm.

List: A data structure containing a single unordered set of properties or elements of the same type. Note: the term “list” is used in other SMPTE documents and might have a different meaning in these.

Manifest document: A representation of the details and state of a transaction. Takes the form of an element.

Mapping: A specification for how MDP messages and data structures shall be represented on the network.

Media Dispatch Protocol: A standard mechanism for two organizations to initiate a delivery, agree on the details of how it should happen, and manage the delivery.

Message: A message passed between agents. May be a command, confirmation, query or reply.

Message endpoint: A URL at which an agent receives a message.

Negotiation: The process by which agents agree the details of the transfers within a transaction.

Organization: A company, corporation, department, etc. that requires delivery of files to or from another organization.

Parameter: Information passed within a message.

Pause: Stop a transfer in such a way that it can be later resumed.

Profile: A specification of a subset of MDP intended to aid interoperability. Defines which messages and data may and may not be used, and defines allowed sequences of messages and actions.

Property: A data structure containing a single simple value.

Pull-mode transfer: A transfer initiated from the receiving end.

Push-mode transfer: A transfer initiated from the sending end.

Query: A message that allows an agent to request information from another agent.

Reply: A message in immediate response to a query.

Resume: Restart a paused transfer.

Target agent: The agent that receives a message to initiate a transaction.

Transaction: The sequence of steps undertaken by two agents while negotiating and performing the transfer of a set of files.

Transfer: File transfer.

Transfer endpoint: A sending or receiving URL used for a file transfer.

6 Background (Informative)

Treating media as files within systems and within organizations is now commonplace, with interoperability aided through standards such as MXF (SMPTE 377M). Furthermore, as high-bandwidth IP-based networks become increasingly cost-effective, delivery of media content as files is becoming an attractive alternative to the physical movement of tapes, films or disks, or the use of video and audio circuits.

A range of mechanisms exist to transfer files, including well-known protocols such as FTP (RFC 959) and HTTP (RFC 2616), secure protocols such as SFTP and HTTPS (RFC 2818), as well as high-performance transfer protocols developed by academic and commercial organizations. These mechanisms can be expected to change with time as better protocols are developed and standardized.

Often, file delivery will be required as part of an automated workflow. For example approval of a program by a producer might trigger initiation of a delivery, and completion of the delivery might cause an asset management system to automatically update its database with details of the content.

To achieve interoperability between systems and/or organizations within an automated workflow, it is important to have standards for formats and metadata, and for network protocols. But it is also necessary to have a standard approach for initiation of a delivery, for negotiation of the file set to be sent, the delivery mechanism and other details of the transfer, and for reporting the progress and outcome of the transaction. The Media Dispatch Protocol provides such a standard.

7 Overview of Specification

7.1 Context for Media Dispatch Protocol (Informative)

Figure 1 shows MDP in the context of two organizations that need to deliver media as files. MDP agents, usually referred to in this standard simply as “agents”, are software entities that manage the deliveries. Typically an agent will provide an API to allow media applications to request deliveries, check on how they are progressing, and be informed of their outcome. The applications accessing the API might range from a simple user interface application to a complex workflow automation system. Where delivery occurs across the Internet or other public network, often an agent will run on a separate computer outside a firewall.

For simplicity, Figure 1 shows file transfer as occurring between the same systems on which the agents are hosted. However, in general this might not be the case. For example MDP supports a scenario where the agent receiving the files performs a pull-mode transfer from a separate server on which the files reside, but which does not itself run an MDP agent.

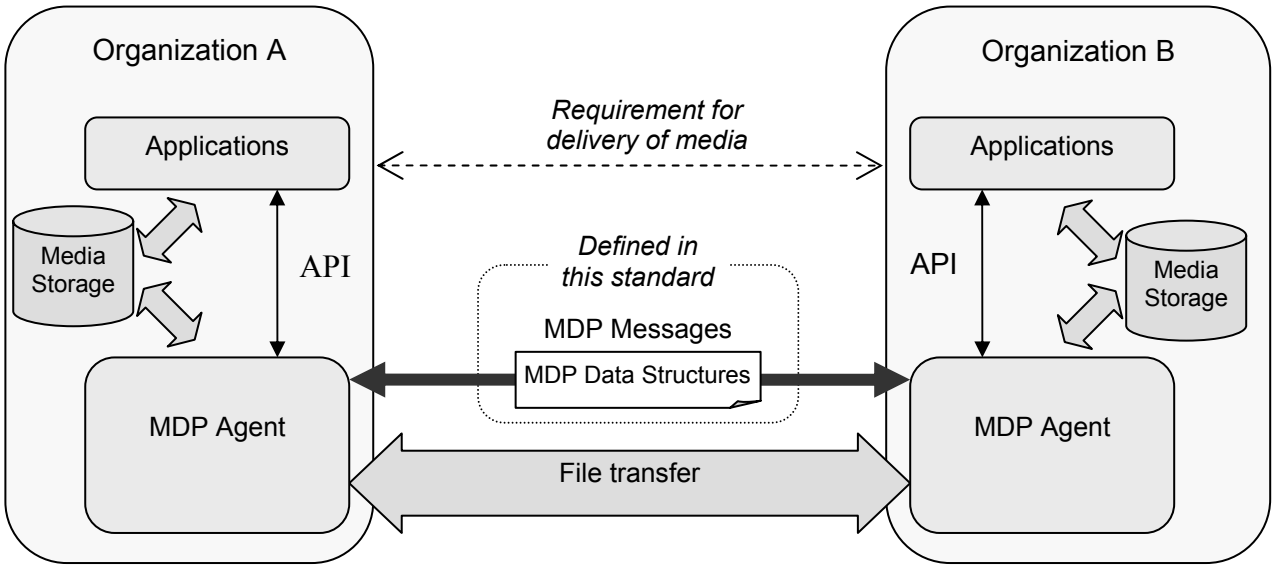

Figure 1 – Context for Media Dispatch Protocol

Agents communicate with each other to agree what files will be transferred, by what mechanism, and when, and to exchange information on the progress and outcome of the transfers. This communication is performed using the Media Dispatch Protocol, and is implemented using a standard set of messages (section 8) containing a standard set of data structures (section 9). The agent that sends the message to initiate a delivery is called the initiator agent; the agent receiving this message is called the target agent.

Note: This standard is not concerned with the details of any API the agent offers to the application layer, nor with the details of how the agent accesses the media storage shown in the figure.

7.2 Organizations and Agents

An agent shall be identified by its message endpoint URL. This shall have type xs:anyURI as defined by the W3C XML Schema Datatypes definition.

An agent acts on behalf of an organization. An organization shall be identified by a text string with type `xs:string`.

7.3 Transactions and Projects

A transaction shall be defined as the sequence of steps undertaken by two agents while negotiating and performing the transfer of a set of files. A transaction may involve any number files (including zero).

A transaction shall be associated with a project. A project shall be identified by a text string, referred to in this standard as `projectid`. This string shall be unique between the organizations involved in the transaction, and shall have type `xs:string`.

Note: An example of a `projectid` might be a program number. In general more than one transaction may have the same `projectid`.

A transaction shall be identified by a text string, referred to in this standard as `transactionid`. Each transaction associated with the same `projectid` shall have a unique `transactionid`. `transactionid` shall have type `xs:string`.

Note: `transactionid` does not have to be globally unique; two transactions may have the same `transactionid` if they are associated with different projects or organizations. A transaction can be identified uniquely on a global basis by combining the organization identifiers, `projectid` and `transactionid`.

7.4 Manifest Document (Informative)

The manifest document is a representation of the details and state of a transaction. It contains information about:

- the transaction itself (organization and agent names, project and transaction identifiers, etc.)
- the files that are offered for transfer, or that have been transferred (name, size, etc.)
- the transfer options proposed or used (transfer protocol, time, which organization is responsible for controlling the transfer, etc.)
- which transfer option(s) are associated with each file.
- the current status of any file transfers within the transaction, for example: rejected, awaiting transfer, in progress, succeeded.

Section 9.4 specifies the data structures within the manifest document. These are summarized in Figure 2.

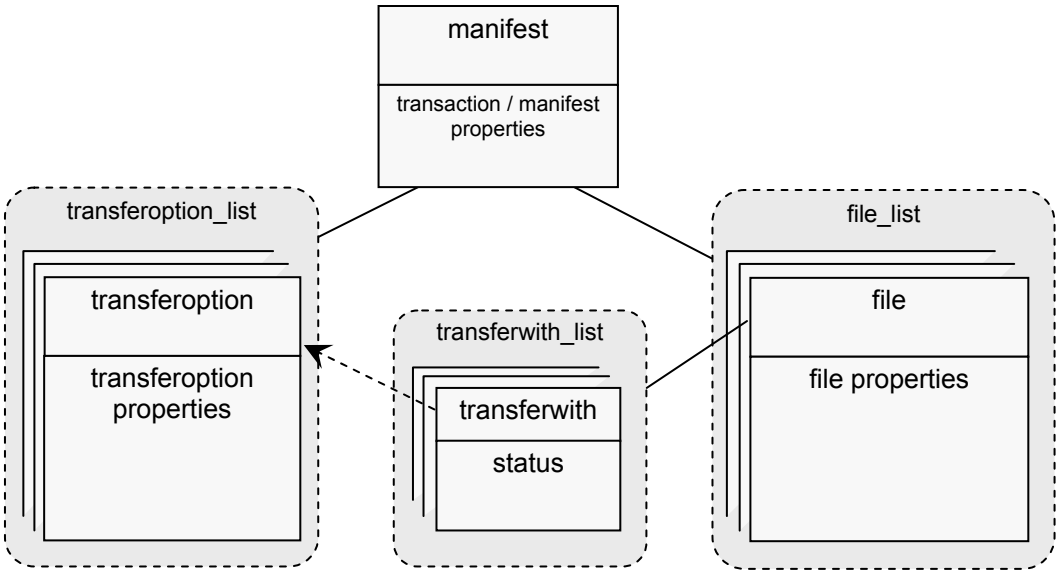

Figure 2 – Summary of Manifest Document Data Structures

Note: The term "`manifest` element" refers to the highest-level data structure within the manifest document.

Note: A `transferwith` element references a `transferoption` element by means of a unique identifier.

An example of a manifest document is given in Annex B.2.

7.5 Transaction Lifecycle (Informative)

The following phases are identified within the lifecycle of a transaction:

In negotiation: The agents send progressively updated versions of the manifest document to agree the details of the transfers. For example the initiator agent can send a manifest document proposing different transfer options and the target agent can respond accepting one of these and rejecting the others.

Awaiting transfer: The negotiation has been successfully completed; no unresolved details, such as the names of the file(s) or the transfer protocol(s) to be used, remain. Both agents have sent manifest documents that indicate that they agree with the current set of details for the transaction, but no file transfer(s) have started yet.

Transfer in progress: At least one file transfer has started, and has not yet finished. Agents can use manifest documents to communicate the state of the transfers.

Completed: All file transfers have completed successfully or unsuccessfully (or no transfers were started). An agent signals this by sending a final manifest document indicating the result of each transfer and the final number of bytes transferred (if known).

Figure 3 shows the phases within the transaction lifecycle. The solid lines correspond to a successful transaction, while the dotted lines correspond to cases where the transaction is terminated, either by choice or because no agreement could be reached on the details of the transfer(s).

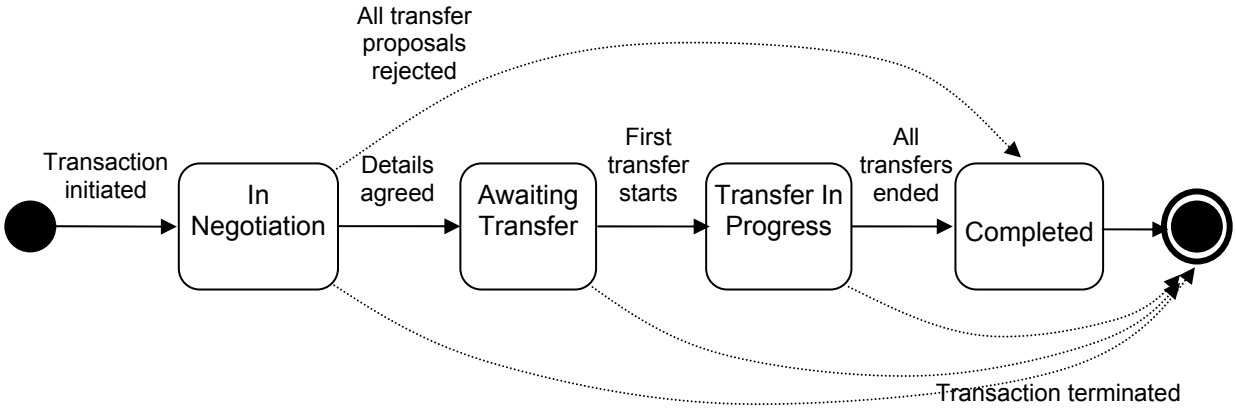

Figure 3 – Transaction Lifecycle

Note: The phases described in this section are not a normative part of this standard.

7.6 Messages (Informative)

Agents communicate using the messages normatively defined in section 8. MDP defines four categories of message:

Commands allow an agent to initiate, complete or terminate a transaction, to send information about the transaction and transfers, and to request that a transfer or transfers be paused or resumed. Commands have names beginning with `cmd_`.

Confirmations provide synchronous positive or negative acknowledgements of commands. Confirmations have names beginning with `cnf_`.

Queries are used to request information from another agent. Queries have names beginning with `qry_`.

Replies provide synchronous answers to queries, or indication of failed queries. Replies have names beginning with `rpl_`.

Each agent exposes a message endpoint, on which it listens for messages.

Note: This standard does not specify a mechanism for an agent to discover another agent's message endpoint for the first time.

An example showing the sequence of messages used in a simple transaction is shown in Annex B.1.

7.7 Transfer (Informative)

The manifest document sent at the end of the negotiation phase of the transaction contains the information required to determine how the transfers will happen. Each file that was accepted for transfer during negotiation will have a transfer protocol, base URL and pathname associated with it. In addition, there can be some optional data that specifies the time of the transfer, any security requirements, and an "order of transfer" priority.

The manifest document also states which organization's agent will control each file transfer. This can be either the initiator or the target agent. This agent will use the protocol, base URL and pathname information in the manifest document to determine the transfer endpoint for each transfer, and will initiate the transfer accordingly. It will then monitor the progress of the transfer.

Files can be transferred in any order, simultaneously or sequentially.

While in the transfer state, an agent can request file transfer progress information from the other agent. It does this by sending a query message, to which the other agent responds with information describing both the current state of transfer (in progress, paused, etc.), and also how much data has been transferred.

The agent that is not controlling the transfer(s) can request that the controlling agent pause or resume of one or more transfers.

7.8 Capabilities Document (Informative)

Support for some parts of this specification is optional. In addition agent implementations might place limits on what can be supported, an example being the maximum number of files in a transaction. An agent can request that another agent send a document, known as the capabilities document, describing the features that it supports, and associated limits. By requesting a capabilities document first, an agent can avoid initiating a transaction that will clearly not be supported by the other agent.

An example of a capabilities document is given in Annex B.3.

7.9 Mappings

An MDP mapping specification shall specify how the message set and data structures shall be represented and transmitted on the network. Messages shall be sent between agents using one of these mappings. An agent shall support at least one MDP mapping specification.

Note: An example of a mapping specification is MDP/XML/HTTP (SMPTE 2032-2), in which the messages and data structures take the form of HTTP request / response pairs and XML elements.

In the event of conflict between normative text in this protocol specification and normative text in a mapping specification, this protocol specification shall take priority.

7.10 Profiles

An MDP profile specification shall define a subset of the protocol specification. A transaction shall comply with one of these profiles. An agent shall support at least one MDP profile specification.

Note: This allows for the possibility of more straightforward agent implementations which provide limited, but well-defined, functionality, as well as more complex implementations, that provide support for a wider range of features. However, in each case, the set of features is defined clearly by the profile, meaning that any two agents supporting the same profile will operate together correctly.

A profile specification shall include: 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.

Note: An example of a profile specification is MDP Basic Target Pull Profile (SMPTE 2032-3), which specifies use of pull-mode transfer and a simple negotiation procedure.

In the event of conflict between normative text in this protocol specification and normative text in a profile specification, this protocol specification shall take priority.

7.11 Dark Metadata

Agents shall not send any information that is not specified in this standard within a message.

Note: This means that “dark metadata” must not be sent within the manifest document. For example, agents must not extend the manifest document to include contract information, or staff email addresses. Dark metadata can however be sent in a file listed in the manifest document, or referenced by means of URLs included in the manifest document.

8 Message Set of Specification

MDP agents shall communicate using the messages defined in this section. Agents shall not communicate using any other messages.

Agents shall comply with any additional requirements on the use of messages specified in any applicable MDP profile specification.

8.1 Parameters

Commands and queries can include parameters to convey information required by the message.

Table 1 defines the parameters that can appear. Agents shall not include any other parameters. Agents shall only use the parameters specified for the particular message (sections 8.2 and 8.4).

Types prefixed with `xs:` are defined by the W3C XML Schema Datatypes definition. Types marked as element, property or list are defined in the section of the data specification specified in brackets.

Table 1 – Parameters for Commands and Queries

Name	Description	Type
`agent`	The message endpoint of the agent sending the message.	`xs:anyURI`
`txorg`	The organization on whose behalf the agent sending the message is acting.	`xs:string`
`rxorg`	The organization on whose behalf the agent receiving the message shall act.	`xs:string`
`projectid`	The project to which this message relates.	`xs:string`
`transactionid`	The transaction to which this message relates.	`xs:string`
`manifest`	Information about the transaction (top-level element of the manifest document).	Element (9.4)
`capabilities`	Information about the capabilities of an agent.	Element (9.11)
`fileid_list`	The files to which this message relates.	List of `fileid` properties (9.7)
`notificationspec_list`	Reserved.	
`notification`	Reserved.	
`abort`	Reason for terminating transaction.	Element (9.16)

8.2 Commands

The following sub-sections define the commands that an agent may send, and shall accept. The name of the command corresponds to the name of the sub-section.

An agent shall send all required parameters with the command. An agent may send any optional parameter.

The agent receiving the command shall respond with one of the listed confirmations.

8.2.1 `cmd_initiatingtransaction`

Description:

The initiator agent shall send this command to the target agent to indicate an intent to create a new transaction.

Required parameters:

`manifest`
`agent`
`txorg`
`rxorg`

Confirmation responses:

`cnf_ok`: The target agent found no syntax errors in the manifest parameter and agrees to creation of the transaction.

Note: This does not guarantee that there are no errors in the `manifest` parameter, other than syntax errors. It is recommended that implementations also check for structural validity and other logical errors in the `manifest` document, for example, the wrong number of `file` elements. In such cases, the agent receiving the command can send a `cnf_error` with a suitable `reason` property, for example `MANIFEST_ERROR`.

`cnf_error`: Some problem occurred, or the target does not agree to the creation of the transaction.

8.2.2 cmd_sendingmanifest

Description:

An agent shall send this command to convey information about the current state of the transaction.

Note: This command is typically used to update the current proposals during the “In Negotiation” phase of the transaction, or to inform the other agent that transfers have started.

Required parameters:

`manifest`
`agent`
`txorg`
`rxorg`

Confirmation responses:

`cnf_ok`: The agent receiving the command found no syntax errors in the `manifest` parameter.

Note: This does not guarantee that there are no errors in the `manifest` parameter, other than syntax errors. It is recommended that implementations also check for structural validity and other logical errors in the `manifest document`, for example, the wrong number of `file` elements. In such cases the agent receiving the command can send a `cnf_error` with a suitable `reason` property, for example `MANIFEST_ERROR`.

`cnf_error`: Some problem occurred.

8.2.3 `cmd_pausetransfers`

Description:

An agent may send this command to request that the other agent pause the transfer of one or more files in the current transaction.

Note: This standard does not define how soon the transfer(s) shall be paused, nor how much more of each file can be transferred before the pause takes effect.

Required parameters:

`projectid`
`transactionid`
`agent`
`txorg`
`rxorg`

Optional parameters:

`fileid_list`: If this parameter is not specified, then the command shall apply to all files in the transaction.

Confirmation responses:

`cnf_ok`: The agent acknowledges the request to pause the transfers.

Note: Sending a `cnf_ok` response does not guarantee that the transfers will be paused. For example, some transfers may already be completed, or an implementation might not support pause functionality for certain transfer protocols. It is recommended that the agent sending the command subsequently send a `qry_requestmanifest` query to determine whether each transfer has paused.

`cnf_error`: Some problem occurred.

8.2.4 `cmd_resumetransfers`

Description:

An agent may send this command to request that the other agent resume the transfer of one or more files in the current transaction that have previously been paused.

Note: This standard does not define how soon the transfer(s) shall be resumed.

Required parameters:

`projectid`

`transactionid`

`agent`

`txorg`

`rxorg`

Optional parameters:

`fileid_list`: If this parameter is not specified, then the command shall apply to all files in the transaction.

Confirmation responses:

`cnf_ok`: The agent acknowledges the request to resume the transfer(s).

Note: Sending a `cnf_ok` response does not guarantee that the transfers will be resumed.. It is recommended that the agent sending the command subsequently send a `qry_requestmanifest` to determine whether each transfer has resumed.

`cnf_error`: Some problem occurred.

8.2.5 `cmd_completingtransaction`

Description:

An agent shall send this command to indicate that all transfers in the transaction have completed (successfully or unsuccessfully).

Required Parameters:

manifest
agent
txorg
rxorg

Confirmation responses:

cnf_ok: The agent receiving the command found no syntax errors in the manifest parameter. The transaction is no longer active. Agents shall send no further messages for this transactionid and projectid.

cnf_error: Some problem occurred. The transaction is still active.

8.2.6 cmd_abortingtransaction

Description:

An agent may send this command to terminate the transaction before it is completed.

Required parameters:

projectid
transactionid
abort
agent
txorg
rxorg

Confirmation responses:

cnf_ok: The transaction is no longer active. Agents shall send no further messages for this transactionid and projectid.

cnf_error: Some problem occurred. The transaction is still active.

8.2.7 cmd_sendingcapabilities

Reserved.

8.2.8 cmd_requestnotification

Reserved.

8.2.9 cmd_sendingnotification

Reserved.

8.3 Confirmations

The following sub-sections define the confirmations that an agent may send, and the data structures that they shall contain. The title of each sub-section is the same as the name of the corresponding confirmation.

8.3.1 `cnf_ok`

Shall contain an `ok` element. An agent shall send this to indicate that a command, including all parameters, has been successfully received, understood, and that the agent agrees to semantic meaning of the command.

8.3.2 `cnf_refused`

Reserved.

8.3.3 `cnf_error`

Shall contain an error element. An agent shall send this to indicate that a problem occurred in receiving, understanding or implementing the command.

8.3.4 `cnf_abort`

Reserved.

8.4 Queries

The following sub-sections define the queries that an agent may send, and shall accept. The name of the query corresponds to the name of the sub-section.

An agent shall send all required parameters with the query. An agent may send any optional parameter.

The agent receiving the query shall respond with one of the listed replies.

8.4.1 `qry_requestmanifest`

Description:

An agent shall send this query to ask the other agent to send a manifest document describing the current state of the transaction.

Required parameters:

`projectid`
`transactionid`
`agent`
`txorg`
`rxorg`

Reply responses:

`rpl_manifest`: The agent receiving the query returns a manifest document describing the current state of the transaction.

`rpl_error`: Some problem occurred.

8.4.2 qry_requestcapabilities

Description:

An agent may send this query to ask another agent to send its capabilities document.

Required parameters:

agent
txorg

Optional parameters:

rxorg
projectid

Reply responses:

rpl_capabilities: The agent receiving the query returns a capabilities document. If either rxorg or projectid was included in the query, the agent should send capabilities information that relates specifically to the organization or project.

rpl_error: Some problem occurred.

Note: The agent sending the query can assume that there is some problem with the other agent, and not initiate a transaction at this time.

8.5 Replies

The following sub-sections define the replies that an agent may send, and the data structures that they shall contain. The title of each sub-section is the same as the name of the corresponding reply.

8.5.1 rpl_manifest

Shall contain a manifest element. An agent shall send this in reply to a qry_requestmanifest.

8.5.2 rpl_capabilities

Shall contain a capabilities element. An agent shall send this in reply to a qry_requestcapabilities.

8.5.3 rpl_delayed

Reserved.

8.5.4 rpl_error

Shall contain an error element. An agent shall send this to indicate that a problem occurred in receiving, understanding or implementing the command.

8.5.5 rpl_abort

Reserved.

9 Data Specification

9.1 Data Structures

The data specification defines the name, type and semantics of the data structures that messages can contain. Messages shall not contain any other data structure.

There are three types of structure:

Elements: An element may contain properties, lists or other elements.

Properties: A property has a single, simple value. Some properties have a restricted set of allowed values, as specified in the data dictionary (section 10).

Lists: A list contains a single unordered set of properties or elements, all of the same type.

Elements shall not contain any element, property or list not defined here, or any other type of content.

Note: In the examples given in this standard, elements, properties and lists are represented as XML elements as specified by the MDP/XML/HTTP mapping specification. However, this is merely informative, and other future mappings might use different representations.

9.2 Character set

All elements, lists, properties and property values shall be representable using characters from the set defined in ISO/IEC 10646.

9.3 Element tables

Each table in sub-sections 9.5 onwards specifies an element. The rows of the table specify the structures (elements, lists and properties) that may appear within the element. The columns specify:

Name: Name of the structure

Description: Description of the meaning of the structure, and any rules or restrictions regarding its usage.

Type: Type of the structure. Where this is prefixed with xs: the structure is a property and the type is defined by the corresponding W3C XML Schema Datatypes definition. A type marked as element is defined in the corresponding section of this data specification.

Ref: A reference to the section of this standard that defines the type of the structure (elements and lists), or specifies the allowed values (properties).

Required (default): If “Yes”, then the structure shall appear in the element. If “No”, then it may be omitted, unless required by an MDP profile specification. A default value may be specified in brackets: an agent reading a message in which this structure is not present shall act as though the structure were present and had this default value.

9.4 Representation of Date and Time

All properties with type `xs:dateTime` shall use Coordinated Universal Time (UTC) and shall use the "Z" time zone indicator as defined in ISO/IEC 8601.

9.5 manifest

Name	Description	Type	Ref	Required (default)
initiator	The organization represented by the initiator agent.	xs:string	7.2	Yes
target	The organization represented by the target agent.	xs:string	7.2	Yes
projectid	The project to which this manifest document refers.	xs:string	7.3	Yes
transactionid	The transaction to which this manifest document refers.	xs:string	7.3	Yes
initiatoragent	Message endpoint of the initiator agent.	xs:anyURI	7.2	Yes
targetagent	Message endpoint of the target agent.	xs:anyURI	7.2	Yes
profile	MDP Profile specification to which the transaction conforms.	xs:string	10.2	No
usecase	Indicates that either the initiator or target agent will initiate all transfers and that all transfers will be push- or pull-mode.	xs:string	10.3	No
transferoption_list	Transfer options that relate to the transaction.	List of transferoption elements	9.6	No
file_list	Files that relate to the transaction.	List of file elements	9.7	No
generation	Reserved.			
comment_list	Comments. Not for programmatic use by MDP agents.	List of comment properties of type xs:string		No
updated	Information about the last update to the manifest element.	updated element	9.10	Yes

9.6 transferoption

Name	Description	Type	Ref	Required (default)
troptid	Identifier for the transferoption. Shall be unique for this manifest element.	xs:string		Yes
direction	Direction of transfer	xs:string	10.6	Yes
protocol	Transfer protocol name.	xs:string	10.5	Yes
sender	Base URL for pull-mode transfers. file pathname properties shall be appended to this to give the transfer endpoint (URL from which the file shall be transferred).	xs:anyURI		No, but see section 10.6
receiver	Base URL for push-mode transfers. file pathname properties shall be appended to this to give the transfer endpoint (URL to which the file shall be transferred).	xs:anyURI		No, but see section 10.6
controller	Organization whose agent will initiate or has initiated the transfer. Shall match either initiator or target in the manifest element.	xs:string		Yes
availablefrom	Transfer will be initiated no earlier than this time.	xs:dateTime		No
availableto	Transfer ought to be completed no later than this time.	xs:dateTime		No
queryprogress	Information about the number of bytes transferred can be provided during transfer.	xs:boolean		Yes
canpause	It is possible to pause the transfer, and later resume the transfer without any loss of data.	xs:boolean		Yes
maxsize	Transfer supports files of at least $2^{maxsize}$-1 bytes.	xs:nonNegativeInteger		Yes
encrypt	Transfer will be encrypted using a mechanism permitted by the profile.	xs:boolean		Yes
authenticate	Transfer will be authenticated using a mechanism permitted by the profile.	xs:boolean		Yes
integritycheck	Transfer will check file integrity using a mechanism permitted by the profile.	xs:boolean		Yes

Name	Description	Type	Ref	Required (default)
comment_list	Comments. Not for programmatic use by MDP agents.	List of comment properties of type xs:string		No
details_list	URLs to external source(s) of additional metadata about the transferoption.	List of details properties of type xs:anyURI		No
updated	Information about the last update to the transferoption element.	updated element	9.10	Yes

9.7 file

Name	Description	Type	Ref	Required (default)
fileid	Identifier for the file. Shall be unique for the projectid specified in the manifest element.	xs:string		Yes
pathname	Pathname of file, relative to the sender or receiver in the relevant transferoption.	xs:string		Yes
size	Size of the file in bytes.	xs:nonNegativeInteger		No, but see note 1.
mimetype	MIME media type and subtype of the file, as defined in RFC 2046.	xs:string		No (application/octet-stream). See also note 2.
filetype	Reserved.			
hash	Hash (digest) information to support integrity checking or unique identification of the file.	hash element	9.8	No
isencrypted	Reserved.			
mustencrypt	Transfer must occur using a transferoption with property encrypt set to true.	xs:boolean		No (true)
mustauthenticate	Transfer must occur using a transferoption with property authenticate set to true.	xs:boolean		No (true)

Name	Description	Type	Ref	Required (default)
mustintegritycheck	Transfer must occur using a transferoption with property integritycheck set to true.	xs:boolean		No (true)
startafter	Transfer must be initiated no earlier than this time.	xs:dateTime		No
finishbefore	Transfer must be completed no later than this time.	xs:dateTime		No
priority	Priority for transfer of file. Lower numbers have higher priority.	xs:integer		No (positive infinity)
bytestransferred	Bytes transferred so far.	xs:nonNegativeInteger		Yes
transferwith_list	Transfer options associated with the file.	List of transferwith elements	9.9	No
comment_list	Comments. Not for programmatic use by MDP agents.	List of comment properties of type xs:string		No
details_list	URL(s) to external metadata about the file.	List of details properties of type xs:anyURI		No
updated	Information about the last update to the file element.	updated element	9.10	Yes

1. Although size is optional (e.g. the file might be created on demand by a transcoding process, so might not yet exist during negotiation), agents should provide this information where known. Some profile specifications might require this information to be present.

2. The MIME type can include any required or optional parameters after the type, separated by semicolons. For example, MXF (SMPTE 377M) has the MIME type application/mxf registered with IANA (RFC 4539). The "ULs" parameter can be used to specify MXF header information in the MIME type, e.g.

```
<mimetype>application/mxf; ULs="urn:oid:1.3.52.4.1.1.1.13.1.2.1.1.1,
urn:oid:1.3.52.4.1.1.1.4.1.2.2.2.1.1"</mimetype>
```

means: *OP1a MXF containing IEC DV video at 25 Mbps, 525/59.94*.

9.8 hash

Name	Description	Type	Ref	Required (default)
hashtype	Type of hash used.	xs:string	10.6	Yes
hashvalue	Hash (digest) value as a hexadecimal string.	xs:hexBinary		Yes

9.9 transferwith

Name	Description	Type	Ref	Required (default)
troptref	Reference to the transferoption. Shall match a troptid property in the same manifest document.	xs:string		Yes
status	Status of the transfer, for example whether it has been accepted or rejected, whether it is in progress, whether it has succeeded.	xs:string	10.8	Yes
updated	Information about the last update to this transferoption element.	updated element	9.10	Yes

9.10 updated

Name	Description	Type	Ref	Required (default)
by	Organization whose agent created or last modified the element. Shall match either initiator or target in the manifest element.	xs:string		Yes
datetime	Time of creation or modification of the element.	xs:dateTime		Yes
updatereason	Why the element was created or modified.	xs:string	10.9	Yes, for a transferwith element whose status property has value rejected, paused, stalled or failed.
comment	Comment. Not for programmatic use by MDP agents.	xs:string		No

9.11 capabilities

Name	Description	Type	Ref	Required (default)
agent	Message endpoint of the agent whose capabilities are described.	xs:anyURI	7.2	Yes
org	Organization to which the capabilities document apply.	xs:string	7.2	Yes
projectid	Project to which the capabilities document apply.	xs:string	7.3	No. (capabilities document shall apply to any project)
profile_list	Supported MDP Profiles.	List of profile properties of type xs:string	10.2	No
usecase_list	Supported logical use cases.	List of usecase properties of type xs:string	10.3	No
mapping_list	Supported mappings of the message set.	List of mapping properties of type xs:string	10.4	No
protocol_list	Supported transfer protocols.	List of protocol properties of type xs:string	10.5	No
direction_list	Supported transfer directions (independent of supported transfer protocols).	List of direction properties of type xs:string	10.6	No
hashtype_list	Supported hash types.	List of hashtype properties of type xs:string	10.7	No
filetype_list	Reserved.			
eventtype_list	Reserved.			
maxmanifestbytes	The largest supported manifest document, in bytes.	xs:nonNegative Integer		No
maxfiles	The maximum number of file elements supported in a manifest document.	xs:nonNegative Integer		No
maxtropts	The maximum number of transferoption elements supported in a manifest.	xs:nonNegative Integer		No

Name	Description	Type	Ref	Required (default)
pause	The agent can pause and resume transfers if requested with cmd_pausetransfers and cmd_resumetransfers, and will signal a paused transfer using the transferwith status property.	xs:boolean		No
stall	The agent will signal a stalled transfer using the transferwith status property.	xs:boolean		No
maxsize	The agent supports transfer of files of at least $2^{maxsize}$-1 bytes.	xs:nonNegative Integer		No
encrypt	The agent supports transfers encrypted by a mechanism permitted by a profile in profile_list.	xs:boolean		No
authenticate	The agent supports transfers authenticated by a mechanism permitted by a profile in profile_list.	xs:boolean		No
integritycheck	The agent supports the checking transfers for file integrity by a mechanism permitted by a profile in profile_list.	xs:boolean		No
notification	Reserved.			
gzipmessage	Reserved.			
partialupdate	Reserved.			
encryptxml	Reserved.			
signxml	Reserved.			

9.12 ok

This element contains no sub-structures.

9.13 error

Name	Description	Type	Ref	Required (default)
reason	Why the error occurred.	xs:string	10.10	Yes
comment	Comment. Not for programmatic use by MDP agents.	xs:string		No

9.14 `refused`

Reserved.

9.15 `delayed`

Reserved.

9.16 `abort`

Name	Description	Type	Ref	Required (default)
`reason`	Why the transaction was terminated.	`xs:string`	10.10	Yes
`comment`	Comment. Not for programmatic use by MDP agents.	`xs:string`		No

10 Dictionary

10.1 Property Value Tables

In the following tables, the strings in the Value columns shall convey the meanings in the Description column.

Unless otherwise stated, properties may be set to a value starting with "`X-`" or "`x-`" to convey a meaning that does not appear in the table. A value that does not appear in the table and does not start with "`X-`" or "`x-`" shall not be used.

10.2 `profile`

Value	Description
`basic target pull`	MDP Basic Target Pull Profile.
`basic target push`	Reserved.
`basic initiator pull`	Reserved.
`basic initiator push`	Reserved.
`basic`	Reserved.
`full`	Reserved.

10.3 `usecase`

Value	Description
`target pull`	The target agent initiates all file transfers. All transfers are pull-mode.
`target push`	The target agent initiates all file transfers. All transfers are push-mode.
`initiator pull`	The initiator agent initiates all file transfers. All transfers are pull-mode.
`initiator push`	The initiator agent initiates all file transfers. All transfers are push-mode.
`sync`	Reserved.

10.4 `mapping`

Value	Description
`XML/HTTP`	According to MDP/XML/HTTP mapping specification (SMPTE 2032-2).
`SOAP/HTTP`	Reserved.

10.5 `protocol` (Administrative Register)

The meanings of the `protocol` property shall exist in an SMPTE Administrative Register, which shall contain a list of defined values for the `protocol` property that may be specified by agents in the manifest document or capabilities document. The Administrative Register shall also contain a description of the protocol, and a reference to a definition of the protocol. The Administrative Register may contain proper names or trademarks.

Unless assigned through the procedures defined in Annex C, all other values used for the `protocol` property shall being with "`X-`" or "`x-`" as described in section 10.1.

The `protocol` register shall be published on the web site of the SMPTE Registration Authority at the following URL:

`http://www.smpte-ra.org/registers/2032-1/2007/MDP/protocol`

The register shall be published in all formats required by all MDP mapping specifications.

Note: The MDP/XML/HTTP mapping specification (SMPTE 2032-2) specifies an XML representation of the register.

Until the next revision of this document, changes to the protocol register shall be published exclusively on line at the SMPTE Registration Authority web site. Users of this standard are encouraged to check the SMPTE-RA web site regularly for updates to the table.

The Effective Date appearing in the second row indicates the date of the most recent modification of the table. When this standard is published or revised, the Effective Date appearing in the online version initially shall be the same as that appearing in this standard. Subsequent to publication of this standard, the Effective Date

value may be used to determine whether changes have occurred in the online register since the last time such a query was made.

An informative copy of the protocol register contents, including all values of the protocol property valid as of the publication date of this document, appears in Table 2.

Table 2 – `protocol` Administrative Register (Informative)

Property Name: `protocol`

Effective Date: Document Publication

Value	Description	Reference
`HTTP`	HTTP/1.1	`www.ietf.org/rfc/rfc2616.txt`
`HTTP/TLS`	HTTP/1.1 over TLS	`www.ietf.org/rfc/rfc2818.txt`
`FTP`	FTP	`www.ietf.org/rfc/rfc0959.txt`
`FTP/TLS`	FTP over TLS	`www.ietf.org/rfc/rfc4217.txt`
`SFTP`	SSH File Transfer Protocol	`tools.ietf.org/wg/secsh/draft-ietf-secsh-filexfer/`
`ANON-FTP`	FTP with login username "`anonymous`"	`www.ietf.org/rfc/rfc0959.txt`
`FLUTE`	File Delivery over Unidirectional Protocol	`www.ietf.org/rfc/rfc3926.txt`
`NFS`	Network File System	`www.ietf.org/rfc/rfc3010.txt`
`SCP`	SSH Copy Protocol	`www.ietf.org/rfc/rfc4251.txt`
`Aspera Scp`	Aspera Scp	`www.asperasoft.com`
`BitTorrent`	BitTorrent	`www.bittorrent.org/protocol.html`
`CIFS`	Common Internet File System	`www.samba.org`
`DF Raptor`	Digital Fountain Raptor	`www.digitalfountain.com`
`FDT`	Fast Data Transfer	`monalisa.cern.ch/FDT/`
`GridFTP`	Protocol Extensions to FTP for the Grid	`www.ggf.org/documents/GWD-R/GFD-R.020.pdf`
`Datacast XD`	Datacast XD	`www.intldata.ca`
`Fazzt`	Kencast Fazzt	`www.kencast.com`
`rsync`	rsync incremental file transfer utility	`samba.anu.edu.au/rsync/`

Value	Description	Reference
SABUL	SABUL	www.rgrossman.com/sabul.htm
FASTCopy	SoftLink FASTCopy	www.softlink.com
SkyScraper	Triveni Digital SkyScraper	www.trivenidigital.com
UDT	UDP-based Data Transfer Protocol	udt.sourceforge.net
UFTP	UDP-based FTP	www.tcnj.edu/~bush/uftp.html
FileCatalyst	Unlimi-Tech FileCatalyst	www.utechsoft.com
Tsunami	Tsunami	www-iepm.slac.stanford.edu/bw/Tsunami.htm
VFER	VFER	vfer.sourceforge.net
ZSYNC	ZSYNC	zsync.moria.org.uk/paper

10.6 direction

The Specify column determines which of the sender or receiver properties shall appear in the transferoption element that contains the direction property.

Value	Description	Specify
PULL	Pull-mode transfer.	sender
PUSH	Push-mode transfer.	receiver

10.7 hashtype

Value	Description	Defined
CRC32	32-bit Cyclic Redundancy Check.	ISO 3309
CRC64	64-bit Cyclic Redundancy Check.	ISO 3309
MD5	128-bit Message Digest Algorithm 5.	RFC 1321
SHA1	160-bit Secure Hash Algorithm.	FIPS 180-2
SHA256	256-bit Secure Hash Algorithm.	FIPS 180-2
SHA384	384-bit Secure Hash Algorithm.	FIPS 180-2
SHA512	512-bit Secure Hash Algorithm.	FIPS 180-2

10.8 `status`

Only the stated values shall be used (values starting with "`X-`" or "`x-`" shall not be used).

Value	Description
`offered`	The agent proposes that the file be transferred by means of the `transferoption` referenced by the `troptref`.
`accepted`	The agent agrees that the file be transferred by means of the `transferoption` referenced by the `troptref`.
`rejected`	The agent does not agree that the file be transferred by means of the `transferoption` referenced by the `troptref`. An agent shall indicate that it does not agree to the transfer of the file (by any means) by setting the `status` property of all `transferwith` elements for the file to `rejected`.
`in progress`	The file is being transferred by means of the `transferoption` referenced by the `troptref`.
`paused`	Transfer of the file has been deliberately paused.
`stalled`	Transfer of the file has stalled due to network congestion or another problem.
`succeeded`	The file has successfully been transferred by means of the `transferoption` referenced by the `troptref`.
`failed`	Transfer of the file was not successful.

Figure 4 shows how this property may change during a transaction:

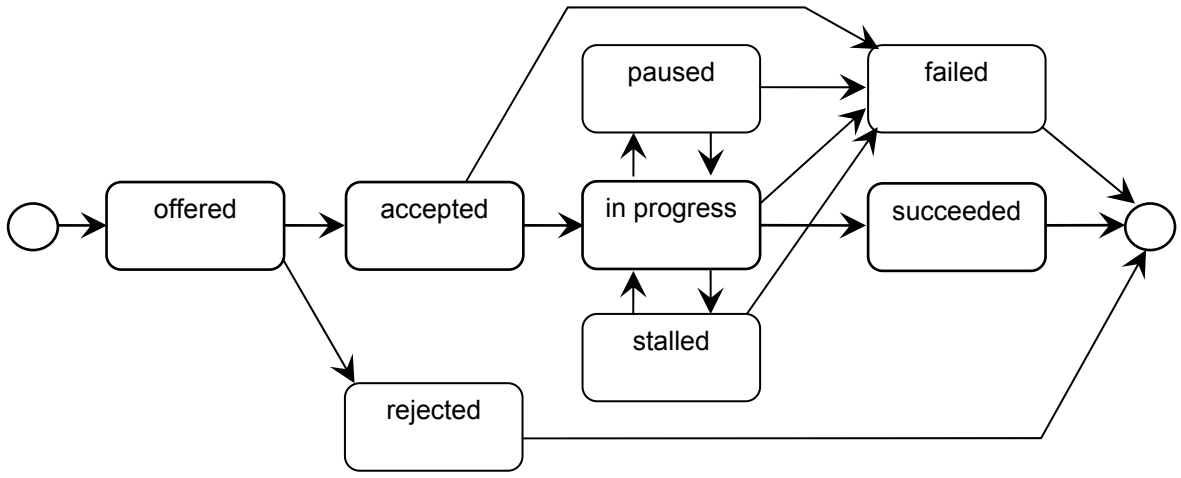

Figure 4 - `status` property

10.9 updatereason

When an agent sets the status property of a transferwith element to rejected, failed, paused or stalled, it shall set the updatereason property to the most appropriate of the following values.

The "Applies to" column indicates those values of the status property for which the updatereason property may be used.

Only the stated values shall be used (values starting with "X-" or "x-" shall not be used).

Value	Description	Applies to
UNSUPPORTED_USECASE	The agent does not support the combination of direction and controller specified in the transferoption. See also the usecase property (section 10.3).	rejected
UNSUPPORTED_PROTOCOL	The agent does not support the protocol specified in the transferoption.	rejected
UNSUPPORTED_DIRECTION	The agent does not support the direction specified in the transferoption.	rejected
UNSUPPORTED_HASH	The agent cannot check the integrity of the transfer using the specified file hash property.	rejected
SUPPORTED_BUT_NOT-_PREFERRED	The agent supports this combination of file and transferoption, but is accepting another transferwith for the file.	rejected
ENCRYPT_NOT_SUPPORTED	The mustencrypt property of the file is true but the encrypt property of the corresponding transferoption is false.	rejected
AUTHENTICATE_NOT-_SUPPORTED	The mustauthenticate property of the file is true but the authenticate property of the corresponding transferoption is false.	rejected
INTEGRITYCHECK_NOT-_SUPPORTED	The mustintegritycheck property of the file is true but the integritycheck property of the corresponding transferoption is false.	rejected
FILE_NOT_REQUIRED	The file is not required.	rejected
FILE_TOO_LARGE	The stated size property of the file is too large.	rejected
TIMES_NOT_POSSIBLE	The agent cannot support the specified combination of transferoption availablefrom and availableuntil properties, and file startafter and finishbefore properties.	rejected
DEADLINE_PASSED	The transfer did not complete successfully before the deadline specified by the file finishbefore property.	failed
REJECTED_BY_OPERATOR	An operator, or external process or system, has requested that the agent reject the transfer.	rejected

Value	Description	Applies to
FILE_NOT_FOUND	The file could not be found using the specified transfer endpoint and pathname information.	failed
TRANSFER_DID_NOT_START	Transfer failed to start for some reason not covered by the above.	failed
TRANSFER_PROCESS_DIED	Transfer started but failed before completion.	failed
INCORRECT_HASH	File received does not have same hash as specified in the file hash property.	failed
INCORRECT_FILE_SIZE	File received does not have same size as specified in the file size property.	failed
UNSATISFIABLE_RANGE	Reserved.	failed
LINK_TOO_SLOW	The file transfer rate is too slow and the agent that initiated the transfer is treating the transfer as having stalled, but not yet failed.	stalled
LINK_TIMED_OUT	The transfer has timed out, and the agent that initiated the transfer is treating it as failed rather than just stalled.	failed
REQUESTED_LOCALLY	The agent that initiated the transfer paused the transfer because of a local request.	paused
REQUESTED_REMOTELY	The agent that initiated the transfer paused the transfer because of a request from the other agent.	paused
INTERNAL_ERROR	An internal error occurred within the agent, or within a system used by the agent.	rejected failed paused

10.10 reason

This property appears in messages that indicate that an agent has detected an error or some other exceptional condition (these messages include cnf_error, rpl_error and cmd_abortingtransaction). The property shall be set to the value whose description is the most appropriate to the reason for sending the message.

Only the stated values shall be used (values starting with "X-" or "x-" shall not be used).

Value	Description
UNKNOWN_PARAM	The message contains one or more parameters that shall not be present.
INVALID_PARAM	One or more message parameters is set to an invalid value.
MISSING_PARAM	One or more required parameters are missing from the message.

Value	Description
BAD_PAYLOAD	A data structure within a parameter is syntactically incorrect. The mapping specification defines when this value should be used.
INVALID_PAYLOAD	A data structure within a parameter is structurally invalid. The mapping specification defines when this value should be used.
INVALID_ID	An `initiator`, `target`, `projectid`, `transactionid` or `fileid` property has an invalid value.
UNSUPPORTED_PROFILE	The agent does not support the MDP profile declared by the `manifest` element's `profile` property.
MANIFEST_TOO_LARGE	The manifest document is too large, for example the total number of bytes is too great or it has too many elements.
MANIFEST_ERROR	The manifest document is valid but is semantically incorrect, for example it contains fewer files than were expected.
NOT_SUPPORTED	The agent does not support the functionality requested.
REQUESTED_BY_OPERATOR	An operator, or an external program or system has requested that the agent send the message.
INTERNAL_ERROR	An internal error occurred within the agent, or within a system used by the agent.

Annex A (Informative)
Bibliography

The following is a Bibliography for the MDP document suite. It does not include the normative references listed in section 2.

SMPTE 377M-2004, Television — Material Exchange Format (MXF) — File Format Specification

SMPTE 2032-2-2007, Media Dispatch Protocol (MDP) — MDP/XML/HTTP Mapping Specification

SMPTE 2032-3-2007, Media Dispatch Protocol (MDP) — Basic Target Pull Profile Specification

SMPTE EG 2032-4-2007 Media Dispatch Protocol (MDP) — Engineering Guideline (Informative)

SMPTE RP 210, Metadata Dictionary Registry of Metadata Element Descriptions

IETF RFC 959, File Transfer Protocol (FTP)

IETF RFC 1867, Form-based File Upload in HTML

IETF RFC 2224, NFS URL Scheme

IETF RFC 2616, Hypertext Transfer Protocol: HTTP/1.1

IETF RFC 2818, HTTP Over TLS

IETF RFC 3010, NFS version 4 Protocol

IETF RFC 3023, XML Media Types

IETF RFC 3926, FLUTE — File Delivery over Unidirectional Protocol

IETF RFC 4217, Securing FTP with TLS

IETF RFC 4251, The Secure Shell (SSH) Protocol Architecture

IETF RFC 4539, Media Type Registration for the SMPTE MXF

IETF Internet Draft: SSH File Transfer Protocol

W3C Extensible Markup Language (XML) 1.1 (Second Edition)

W3C XML Schema 1.1 — Part 1: Structures

Annex B (Informative)
Examples

B.1 Example of message sequence

Figure B.1 shows an example of the messages sent in a transaction, in which a simple two-stage negotiation is used, and the target agent initiates the transfer of files using a pull-mode protocol. It is assumed that the initiator agent already knows the message endpoint of target agent. Note: Such a transaction complies with the Basic Target Pull Profile.

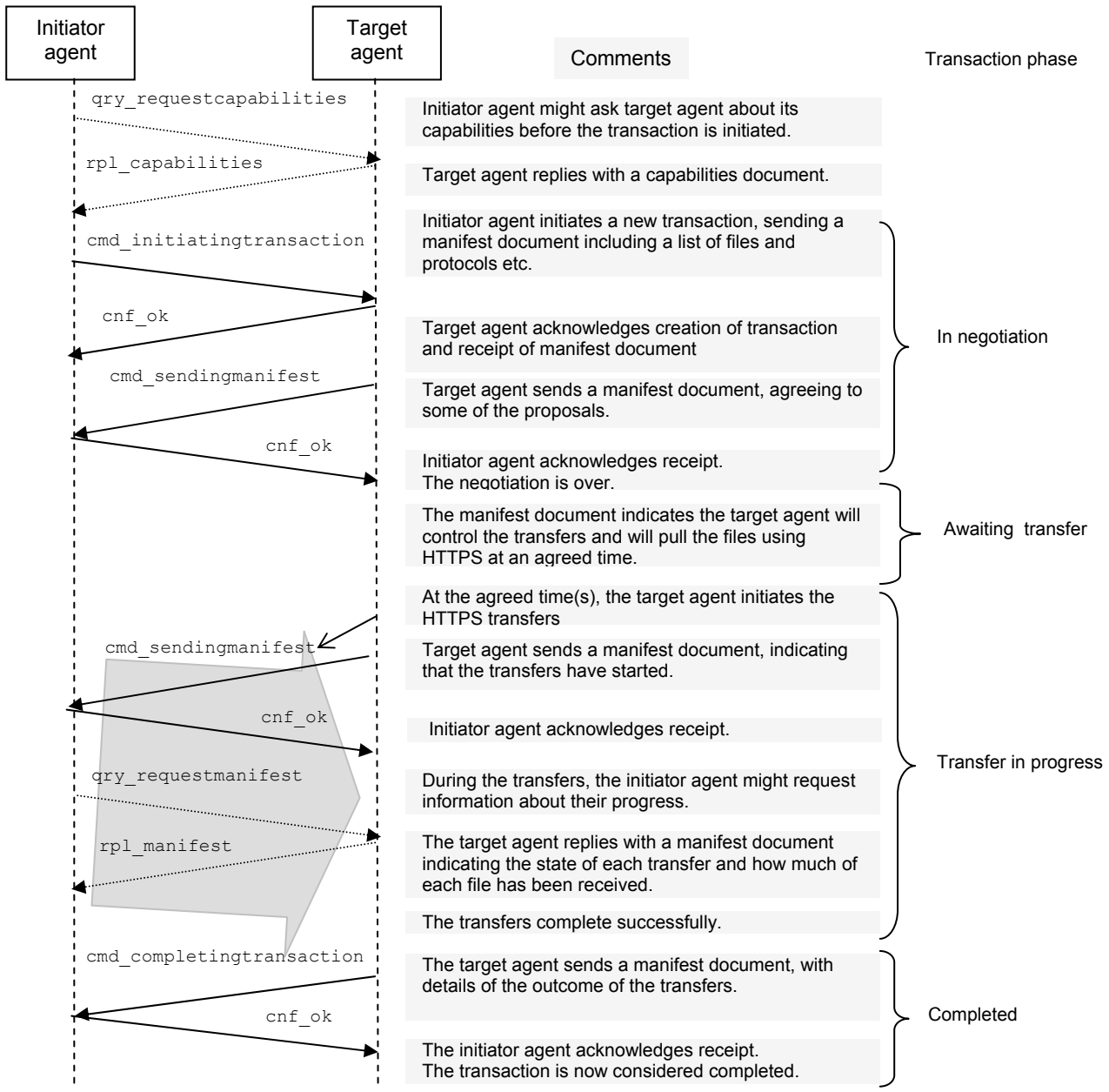

Figure B.1 – Example of MDP Transaction

B.2 Example Manifest Document

Below is shown an example of a simple manifest document that might be sent in a `cmd_initiatingtransaction` message such as in the example of Figure B.1 It mentions six files, and two different transfer options.

Note: This example is shown in XML form as specified by the MDP/XML/HTTP mapping specification (SMPTE 2032-2).

```
<manifest xmlns="http://www.smpte-ra.org/schemas/2032-2/2007/MDP"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.smpte-ra.org/schemas/2032-2/2007/MDP mdp.xsd">
      <transactionid>0123456789abcdef</transactionid>
      <initiator>facility.com</initiator>
      <target>broadcaster.com</target>
      <projectid>P0123456</projectid>
      <profile>basic target pull</profile>
      <initiatoragent>https://facility.com/mdp</initiatoragent>
      <targetagent>https://broadcaster.com/mdp</targetagent>
      <usecase>target pull</usecase>
      <comment list>
            <comment>MDP Manifest document</comment>
      </comment_list>
      <updated>
            <by>facility.com</by>
            <datetime>2007-07-06T19:15:00Z</datetime>
            <comment>initial manifest</comment>
      </updated>
      <!--
       List of options offered for transferring Files
      -->
      <transferoption list>
            <transferoption>
                  <troptid>HTTPS GET from mdp.facility.com</troptid>
                  <direction>PULL</direction>
                  <protocol>HTTP/TLS</protocol>
                  <availablefrom>2007-07-06T19:30:00Z</availablefrom>
                  <availableuntil>2007-07-06T22:00:00Z</availableuntil>
                  <sender>https://mdp.facility.com/transfer/P0123456</sender>
                  <controller>broadcaster.com</controller>
                  <maxsize>48</maxsize>
                  <encrypt>true</encrypt>
                  <authenticate>true</authenticate>
                  <integritycheck>true</integritycheck>
                  <queryprogress>true</queryprogress>
                  <comment_list>
                        <comment>HTTPS GET from mdp.facility.com</comment>
                  </comment list>
                  <updated>
                        <by>facility.com</by>
                        <datetime>2007-07-06T19:15:00Z</datetime>
                        <comment>offered HTTPS GET transferoption</comment>
                  </updated>
            </transferoption>
            <transferoption>
                  <troptid>HTTP GET from mdp.facility.com</troptid>
                  <direction>PULL</direction>
                  <protocol>HTTP</protocol>
                  <availablefrom>2007-07-06T19:30:00Z</availablefrom>
                  <availableuntil>2007-07-06T22:00:00Z</availableuntil>
                  <sender>http://mdp.facility.com/transfer/P0123456</sender>
                  <controller>broadcaster.com</controller>
                  <maxsize>48</maxsize>
                  <encrypt>false</encrypt>
                   ...
            </transferoption>
      <transferoption list>
      <!--
```

```
        List of files offered for transfer
       -->
       <file list>
              <file>
                     <fileid>20031120140000001001</fileid>
                     <pathname>prime/program.mxf</pathname>
                     <size>123456789</size>
                     <mimetype>application/mxf;
                      ULs="urn:oid:1.3.52.4.1.1.1.13.1.2.1.1.1,
                           urn:oid:1.3.52.4.1.1.1.4.1.2.2.2.1.1"</mimetype>
                     <hash>
                            <type>SHA-1</type>
                            <value>3ba8de9ce74bed02bd75023471efe90024352b7a</value>
                     </hash>
                     <isencrypted>false</isencrypted>
                     <mustencrypt>true</mustencrypt>
                     <mustauthenticate>true</mustauthenticate>
                     <mustintegritycheck>true</mustintegritycheck>
                     <priority>1</priority>
                     <bytestransferred>0</bytestransferred>
                     <!--
                      List of transferoptions offered for this file
                     -->
                     <transferwith_list>
                            <transferwith>
                                   <troptref>HTTPS GET from mdp.facility.com</troptref>
                                   <status>offered</status>
                                   <updated>
                                          <by>facility.com</by>
                                          <datetime>2007-07-06T19:15:00Z</datetime>
                                          <comment>offered transfer</comment>
                                   </updated>
                            </transferwith>
                            <transferwith>
                                   <troptref>HTTP GET from mdp.facility.com</troptref>
                                   <status>offered</status>
                                   ...
                            </transferwith>
                     </transferwith list>
                     <details list>
                            <details>https://mdp.facility.com/details/program.mxf.xml</details>
                     </details_list>
                     <comment_list>
                            <comment>Star Wars Episode 7, version 1.0</comment>
                            <comment>16:9, browse quality MXF OP1a
                                                MPEG2 MP@ML AES DMS1</comment>
                     </comment list>
                     <updated>
                            <by>facility.com</by>
                            <datetime>2007-07-06T19:15:00Z</datetime>
                            <comment>offered file</comment>
                     </updated>
              </file>
              <file>
                     <fileid>20031120140000001002</fileid>
                     <pathname>proxies/program.mxf</pathname>
                     <size>2478936</size>
                     ...
              </file>
              <file>
                     <fileid>20031120140000001003</fileid>
                     <pathname>roughcuts/program.aaf</pathname>
                     ...
              </file>
              <file>
                     <fileid>20031120140000001004</fileid>
                     <pathname>rights/program.pdf</pathname>
                     ...
              </file>
              <file>
                     <fileid>20031120140000001005</fileid>
```

```
                        <pathname>subtitles/program.xml</pathname>
                        ...
                </file>
                <file>
                        <fileid>20031120140000001007</fileid>
                        <pathname>metadata/program.xml</pathname>
                        ...
                </file>
        </file list>
</manifest>
```

B.3 Example Capabilities Document

The following is an example of a relatively simple capabilities document represented as XML:

```
<capabilities manifest xmlns="http://www.smpte-ra.org/schemas/2032-2/2007/MDP"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.smpte-ra.org/schemas/2032-2/2007/MDP mdp.xsd">
        <agent>https://facility.com/mdp</agent>
        <org>facility.com</org>
        <profile_list>
                <profile>basic target pull</profile>
        </profile list>
        <usecase list>
                <usecase>target pull</usecase>
        </usecase list>
        <mapping_list>
                <mapping>XML/HTTP</mapping>
        </mapping list>
        <maxmanifestbytes>5000000</maxmanifestbytes>
        <maxfiles>5000</maxfiles>
        <maxtropts>100</maxtropts>
        <maxsize>40</maxsize>
</capabilities>
```

Annex C (Normative)
Register Management

The register established by this document shall be published and maintained in accordance with the procedures defined in this Annex and with the provisions of the SMPTE Administrative Guideline on Registers.

C.1 `protocol` property – dynamic provisions

Extensions to and publication of the `protocol` Administrative Register, an informative copy of which, as of the time of publication of this standard, is included herein in Table 2, shall be according to the procedures provided in Table C.1.

Table C.1 – Procedure for dynamic provisions of `protocol` property

Entities permitted to request extensions	Any entity with an interest in the field.
URL for web browser access	`http://www.smpte-ra.org/registers/2032-1/2007/MDP/protocol`
SMPTE Committee assigned for maintenance	Not required; this is an Administrative Register.
Deletion provisions	Register entries (table rows) shall not be deleted.
Entity permitted to request updates to a listing	The entity listed by SMPTE headquarters as responsible for maintenance of the associated listing, which initially shall be the requesting entity.
Entries permitted to be updated	Value column of existing entries shall not be updated. Description or reference column of existing entries may be updated.
Entity permitted to change responsible entity	Entity currently listed as responsible for keeping registration information up-to-date, or if such entity no longer exists, the technology committee that originally approved this Standard.
Table extension criteria	Compliance with Section 6.6 of Section XIII of SMPTE Administrative Practices (Criteria for Engineering Documents).
Required requesting entity information	Entity name, address, contact person name, telephone & fax numbers and e-mail address of contact person.
Requirements for inclusion in register	The proposed value shall not have been assigned previously in the register and shall not start with either "`x-`" or "`X-`". The proposed description shall be unique. The proposed reference need not be unique. Blank entries for value, description and reference shall not be permitted. A completed application form shall be on file at SMPTE Headquarters.
Information for publication in publicly accessible portion of register	Value of `protocol` property. Brief description of transfer protocol. Reference to full description or specification.
Other information to be included in register management information	Information required in this Table C.1, in the submission form, and as determined necessary for management of the register by SMPTE Headquarters staff

C.2 SMPTE Headquarters Office Procedure

Step 1 – SMPTE headquarters shall make available the submission form shown in Annex D to any entity requesting one or more additional entries (rows) to the register described in this Standard.

Note: it is recommended that this form be made available on the SMPTE-RA website at a URL that easily can be found by an entity examining the online version of the register.

Step 2 – A requesting entity shall complete all parts of the form and return the completed form to SMPTE headquarters.

Step 3 – SMPTE headquarters shall determine that all portions of the submission form have been completed. If any portion of the submission form is not completed, SMPTE headquarters shall notify the requesting entity and return the entire submission form, indicating the information that must be provided.

Step 4 – SMPTE headquarters shall determine that the submission is in compliance with the provisions listed in Annex C.1 above. If any portion of the submission form is not in compliance, SMPTE headquarters shall notify the entity and return the entire submission form, indicating the information that is not in compliance.

Step 5 – Upon determination that all required information has been provided and is in compliance with the requirements of Annex C.1, SMPTE headquarters shall add the requested entry or entries to the protocol register. SMPTE headquarters shall set the effective date in the register to the date the register is uploaded to the SMPTE-RA web site. SMPTE headquarters shall post the revised register on the SMPTE-RA web site in all formats required by MDP mapping specification documents.

Note: The MDP/XML/HTTP mapping specification (SMPTE 2032-2) requires an XML-based representation.

Step 6 – SMPTE headquarters shall notify the requesting entity that this process has been completed. SMPTE headquarters shall retain the information from the submission form about the entity, and shall consider the entity to be responsible for keeping information about the new entry or entries up-to-date.

Annex D (Normative)
Administrative Register Submission Form

Request for extension to MDP protocol property register (http://www.smpte-ra.org/registers/2032-1/2007/MDP/protocol)	
Please return completed request to: SMPTE Registration Authority LLC 3 Barker Avenue White Plains, NY 10601 USA Tel: +1 914 761 1100 Fax: +1 914 761 3115 email: mhyman@smpte.org	
Please complete all the following information about the company or other entity requesting the extension:	
Name	
Address	
Contact name	
Telephone	
Fax	
email	
Preferred method of communication (email / fax / post)	
Please complete one row below for each requested new protocol (add additional rows as required). Please complete all columns.	

Proposed new value for `protocol` property (e.g. "`XYZ-FTP`")	Proposed description (e.g. "XYZ File Transfer Protocol")	Proposed Reference (e.g. "`www.xyzft.com/ftp`")