

SMPTE STANDARD

Media Dispatch Protocol (MDP) —
Protocol Specification



Table of Contents	Page
Foreword	4
Introduction	4
1 Scope	5
2 Conformance Notation	5
3 Normative References	5
4 Acronyms (Informative)	6
5 Definitions	7
6 Background (Informative)	8
7 Overview of Specification	8
7.1 Context for Media Dispatch Protocol (Informative)	8
7.2 Organizations and Agents	9
7.3 Transactions and Projects	9
7.4 Manifest Document (Informative)	10
7.5 Transaction Lifecycle (Informative)	11
7.6 Messages (Informative)	11
7.7 Transfer (Informative)	12
7.8 Capabilities Document (Informative)	12
7.9 Mappings	12
7.10 Profiles	13
7.11 Dark Metadata	13
8 Message Set Specification	13
8.1 Parameters	13
8.2 Commands	14
8.2.1 cmd_initiatingtransaction	14
8.2.2 cmd_sendingmanifest	15

8.2.3 cmd_pausetransfers.....	15
8.2.4 cmd_resumetransfers.....	16
8.2.5 cmd_completetransaction.....	16
8.2.6 cmd_abortingtransaction.....	17
8.2.7 cmd_sendingcapabilities.....	17
8.2.8 cmd_requestnotification.....	17
8.2.9 cmd_sendingnotification.....	17
8.3 Confirmations.....	18
8.3.1 cnf_ok.....	18
8.3.2 cnf_refused.....	18
8.3.3 cnf_error.....	18
8.3.4 cnf_abort.....	18
8.4 Queries.....	18
8.4.1 qry_requestmanifest.....	18
8.4.2 qry_requestcapabilities.....	19
8.5 Replies.....	19
8.5.1 rpl_manifest.....	19
8.5.2 rpl_capabilities.....	19
8.5.3 rpl_delayed.....	19
8.5.4 rpl_error.....	19
8.5.5 rpl_abort.....	19
9 Data Specification.....	20
9.1 Data Structures.....	20
9.2 Character Set.....	20
9.3 Element Tables.....	20
9.4 Representation of Date and Time.....	20
9.5 manifest.....	21
9.6 transferoption.....	22
9.7 file.....	23
9.8 hash.....	25
9.9 transferwith.....	25
9.10 updated.....	25
9.11 capabilities.....	26
9.12 ok.....	27
9.13 error.....	27

9.14 refused.....	28
9.15 delayed.....	28
9.16 abort	28
10 Dictionary	28
10.1 Property Value Tables	28
10.2 profile.....	28
10.3 usecase.....	29
10.4 mapping.....	29
10.5 protocol (Administrative Register)	29
10.6 direction.....	31
10.7 hashtype.....	31
10.8 status	32
10.9 updatereason	33
10.10 reason	34
Annex A Bibliography (Informative)	36
Annex B Examples (Informative)	37
B.1 Example of Message Sequence.....	37
B.2 Example Manifest Document.....	38
B.3 Example Capabilities Document.....	40
Annex C Register Management (Normative).....	41
C.1 protocol Property – Dynamic Provisions	41
C.2 SMPTE Headquarters Office Procedure	42
Annex D Administrative Register Submission Form (Normative)	43

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.

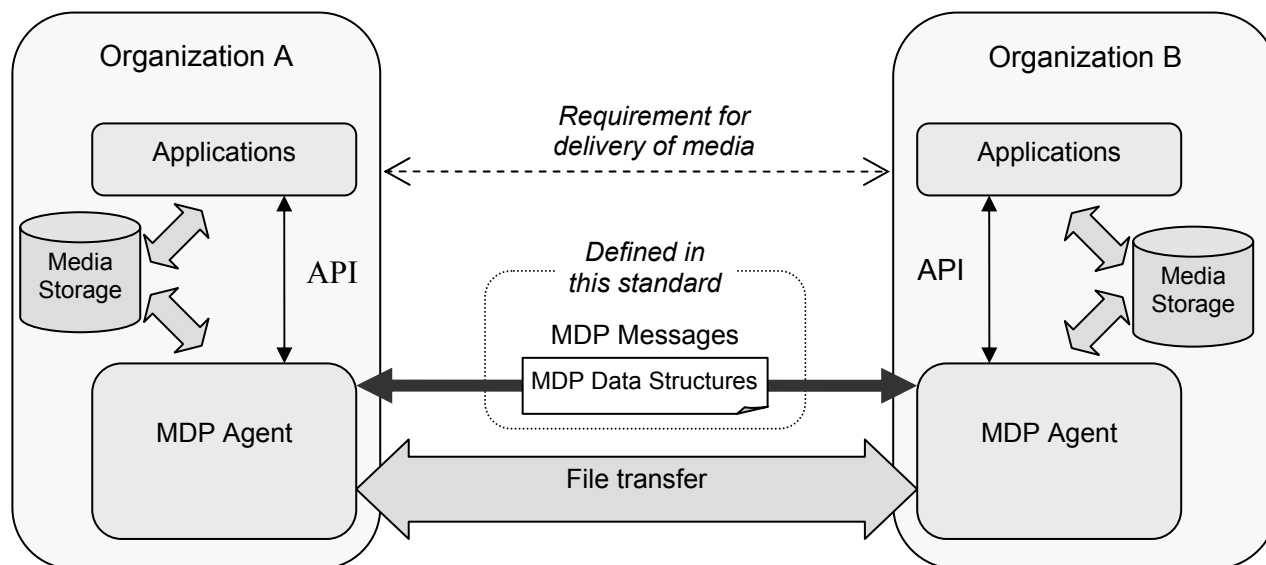


Figure 1 – Context for Media Dispatch Protocol

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.

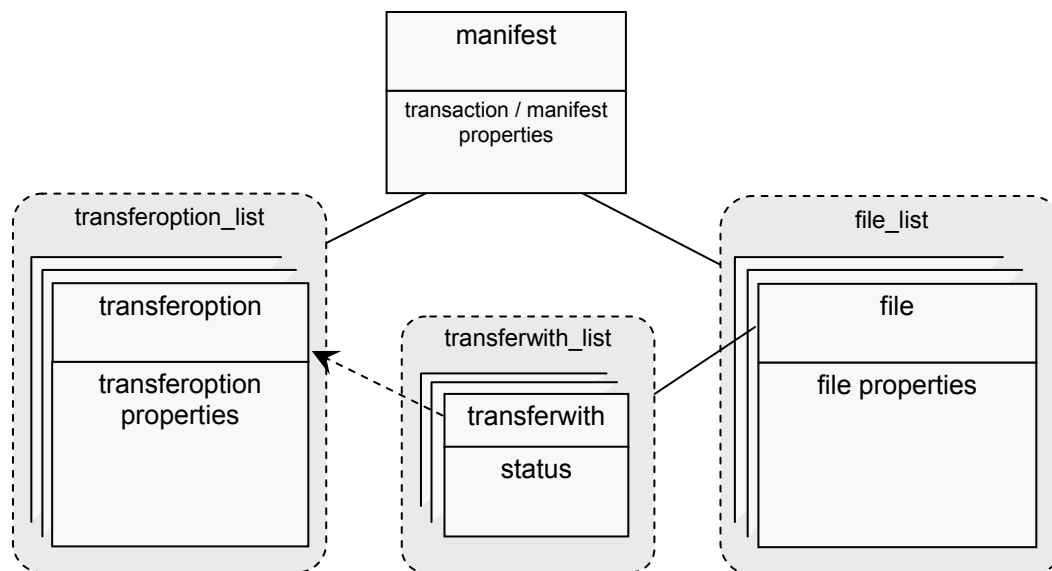


Figure 2 – Summary of Manifest Document Data Structures

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).

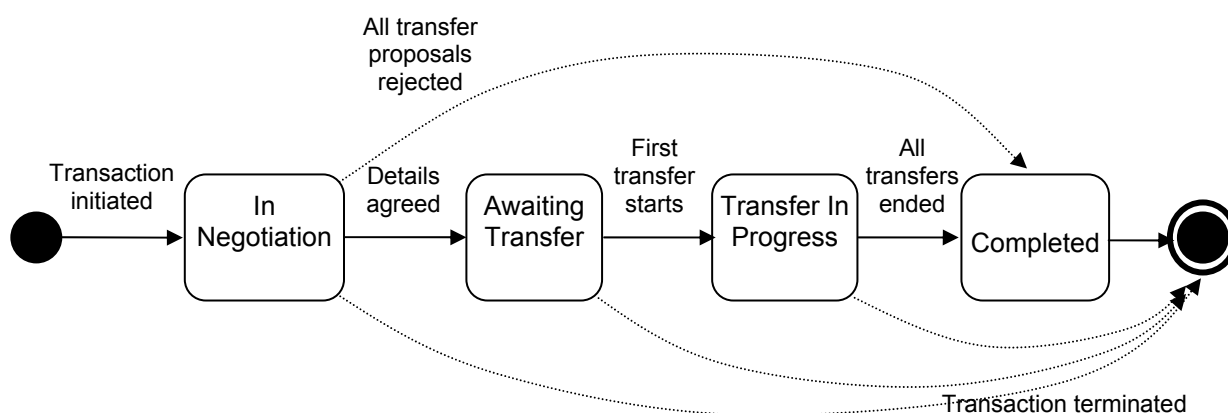


Figure 3 – Transaction Lifecycle

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.