SMPTE RP 2021-9:2012
Revision of RP 2021-9-2009

SMPTE RECOMMENDED PRACTICE

Implementing Broadcast Exchange Format (BXF)

Table of Contents

Approved
March 14, 2012

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 RP 2021-9 was prepared by Technology Committee 34CS.

Intellectual Property

At the time of publication no notice had been received by SMPTE claiming patent rights essential to the implementation of this Recommended Practice. However, attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. SMPTE shall not be held responsible for identifying any or all such patent rights.

Introduction

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

This practice describes implementation recommendations for applications that plan to support SMPTE 2021, commonly known as Broadcast Exchange Format (BXF) version 2.0. The BXF protocol is a sophisticated language that allows different application types (e.g., traffic, automation, etc.) to share information in a standardized format. The standard leaves many of the implementation details to the applications that are utilizing it. For example:

- Applications are only required to implement the aspects of the protocol that are relevant to the type of integration they are providing.

- Content may be identified through an ISAN, house number and/or 1-N alternate identifiers.

This "gray area" makes it possible for applications to implement BXF solutions that, while compliant with the standard, may not be compatible with other BXF implementations. This recommended practice strives to address this ambiguity by identifying common implementation recommendations that can be shared amongst applications that intend to utilize the BXF protocol to exchange information.

1 Scope

This recommended practice has the following objectives:

- Describe the types of messages that will be communicated amongst applications and their corresponding formats within BXF SMPTE 2021.
- Describe the aspects of the BXF protocol which are recommended that applications support to achieve maximum interoperability and also to identify those aspects of the protocol that are not recommended at this time.
- Describe the approaches for representing errors within BXF messages.
- Describe how XML schema extensions should be implemented to maximize interoperability between systems.
- Describe the following extensions:
    - Usage attribute extension
    - Filename element extension
    - Configuration extension

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.

Unless otherwise specified the order of precedence of the types of normative information in this document shall be as follows. Normative prose shall be the authoritative definition. Tables shall be next, followed by formal languages, then figures, and then any other language forms.

3 Normative References

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

SMPTE ST 2021-1:2012, Broadcast Exchange Format (BXF) – Requirements and Informative Notes

RFC 4122, A Universally Unique Identifier (UUID) URN Namespace, IETF, 5 July 2005

4 Note on Wellformedness and Validation (Informative)

Two approaches for determining the integrity of an XML document are wellformedness and validation. According to W3C: *A "well-formed" XML document is defined as an XML document that has correct XML syntax. This means: XML documents must have a root element; XML elements must have a closing tag; XML tags are case sensitive; XML elements must be properly nested; XML attribute values must always be quoted. This should not be confused with a valid XML document, which is defined as a "well-formed" XML document which also conforms to the rules of a Document Type Definition (DTD) or an XML Schema (XSD), which W3C supports as an alternate to DTD*.

The XML schemas that comprise the BXF protocol utilize a “lax” processing model for extensions (i.e., PrivateInformation). Therefore, extensions that have an associated XML schema will validate when a message is processed, whereas extensions without schemas must be wellformed, but will not be validated.

5 Processing Model (Normative)

The processing model describes the rules for handling elements and attributes within BXF messages and extensions to those messages. The following rules shall apply to this practice:

- Ignore elements and attributes that are recognized, but not supported by an implementation. For example, if a house number and alternate identifiers are provided to uniquely identify content and a message consumer only recognizes house number, then the alternate IDs should be ignored, as opposed to rejecting the message as invalid.

- Ignore elements and attributes that are specified within the extensible areas of the protocol (i.e., PrivateInformation) and are not recognized by the message consumer.

6 General Conventions (Normative)

The following conventions shall apply to all BXF messages:

- All element and attribute extensions to the BXF protocol shall have associated XML schemas (and namespaces) so that they can be uniquely identified and resolved.
- It is recommended that systems should utilize a single, shared repository for BXF schemas and any extensions.
- Message consumers shall accept or reject a received message in its entirety. For example, if half of the events on a playlist message can be accepted and the other half cannot, the entire message should be rejected. This convention does not prohibit applications from accepting messages with warnings, where certain values, such as a description, are truncated due to field size restrictions. In this case, the entire message should still be accepted.

- Optional elements and attributes that are not covered within this document may be included within any message. These values may be consumed by applications that understand them, but they are not required to process a message.
- References to GUID in SMPTE 2021 and in this recommended practice shall be interpreted as UUID, as specified in RFC 4122.
- All date/time elements and attributes within a message shall be explicitly or implicitly represented using canonical UTC.
- Messages shall be processed in time sequence where message order is retained between a client and server.
- Message elements shall be processed in the order/sequence that they are specified.
- A message may contain multiple instances of the same element (i.e., same unique identifier). For example, a schedule event is added and updated within an application. The associated changes may be combined into a single event element with an action of add, or specified using two event elements, with the add element followed by the update element, where both elements have the same unique identifier. All elements shall be processed in the specified sequence.
- **BxfMessage/@ext:usage:** The usage extension attribute shall be included on every BXF message. It identifies the payload of the message in a contextual manner so the message consumer can process it in the proper context. Refer to Annex B for a list of recognized usages as well as conventions for creating additional usages. Note that usage values are not configuration types and are not mapped or configured at runtime.
- **BxfMessage/BxfData/PrivateInformation/ext:Filename**: The filename extension element is included in a BXF data message when the associated data is provided in a file form instead of the body of the message. This approach may be utilized when the amount of data is exceedingly large and chunking is not desired.
- Message length shall not exceed 2 MB. Applications should use internal algorithms to separate data into multiple messages. For example, if a user selects a month's worth of schedule information to send, then the application should break the schedule information into multiple messages based on the date or some other criteria.

6.1 BxfMessage Attributes

- **BxfMessage/@id:** The id attribute is required by the BXF schema.
- **BxfMessage/@dateTime:** The date/time attribute is required by the BXF schema. It identifies when the message was created.
- **BxfMessage/@messageType:** The message type attribute is required by the BXF schema. It identifies the message type/category (e.g., request, reply, information, etc.).
- **BxfMessage/@origin:** The origin attribute is required by the BXF schema. It identifies the message originator. This value shall be defined at runtime as an aspect of configuration.
- **BxfMessage/@originType:** The origin type attribute is required by the BXF schema. It identifies the originator's type (e.g., Traffic system). If a valid type value does not exist in the ATSC code point registry, then a value of "Data_server" shall be used by default.
- **BxfMessage/@userName:** The user name attribute is required by the BXF schema. It identifies the user that made the application changes which resulted in the message being created. If the message was generated by a background process or service, then the process or service name shall be specified. For acknowledgement and reply messages that are typically generated automatically, the username that is specified on the original message shall be utilized unless the original message required manual processing by a different user.

- **BxfMessage/@status:** Per the BXF protocol documentation, the status attribute shall be provided on acknowledgement and reply messages.
- **BxfMessage/@error:** The error attribute shall be included if a message is rejected by a receiving system.
- **BxfMessage/@errorDescription:** The error description attribute shall be included if the message is rejected by a receiving system. These conventions are described later in this document.

All other attributes on the BxfMessage element are not required and may not need to be specified.

6.2 Action Oriented Messages

The BXF protocol allows message creators to specify the action (or actions) that are associated with a message. The action attribute exists on most elements within the protocol.

Use of action attributes should be limited to the **BXfMessage/BxfData** element structure, where the message type is either "Request" or "Information".

The overarching action for a message shall be specified so that message consumers will have a common location for identifying the action. The overarching action should be specified on the sub-elements of the BxfData element. Examples:

- **BxfMessage/BxfData/Content/@action**
- **BxfMessage/BxfData/Format/@action**
- **BxfMessage/BxfData/Schedule/@action**

This provides the capability for a single message to have multiple sub-elements with different actions. Such as a content message with multiple Content elements with add, update and remove actions.

If there is only one sub-element, or all of the sub-elements have the same action, then the action may be specified on the BxfData element:

- **BxfMessage/BxfData/@action**

Additional actions may be specified within the message as described within the BXF protocol documentation.

6.3 Query Oriented Messages

Query messages shall be utilized for the following purposes:

- To retrieve information from another application during initial setup/configuration, as well as to synchronize the data within applications when an out-out-sync condition occurs. For example, a traffic system queries a program management system (or vice-versa) to obtain Formats so that the format information within the applications can be synchronized.
- To retrieve detailed information, or information that is required but not stored in an application. For example, a traffic system queries an automation system to obtain timing information on program content which is not stored in the traffic system.

Conventions – Query Messages

- **BxfMessage/BxfQuery/@requestChunking:** The request chunking attribute determines if chunking is desired. Chunking may result regardless of whether it was requested.

- **BxfMessage/BxfQuery/@memoryLimit:** The memory limit attribute identifies the desired maximum message size on a query reply
- **BxfMessage/BxfQueryResponse/@chunkingFlag:** The chunking flag attribute determines if a query reply is chunked.
- **BxfMessage/BxfQueryResponse/@itemNumber:** The item number attribute identifies the number of the current chunk.
- **BxfMessage/BxfQueryResponse/@totalItems:** The total items attribute identifies the final chunk. When total items attribute equals the item number attribute it is final chunk. Note that the total items attribute may be provided on every chunk or only on the final chunk.
- **BxfMessage/BxfQueryResponse/ext:Filename:** The filename extension element is included in a BXF query response message when the requested data is provided in a file form instead of the body of the reply. This approach may be utilized when the response is exceedingly large and chunking is not desired.

6.4 Acknowledgment Messages

Per the BXF protocol documentation, applications that support asynchronous messaging in a connected environment shall acknowledge all information, request and heartbeat messages that are received.

Example messages:

Successful Acknowledgement
Warning Acknowledgement
Error Acknowledgement
Error Acknowledgement (malformed XML)

Conventions – Acknowledgment Messages

BxfMessage/@ext:usage: The usage shall be “Application Acknowledgment”.

BxfMessage/@messageType: The message type shall be “Acknowledgement”.

BxfMessage/@destination: Acknowledgement messages shall include a destination, which is the origin in the request message.

BxfMessage/@status: Acknowledgement messages shall include a status indicating if the original message was accepted or rejected.

BxfMessage/@action: Acknowledgement messages should not contain an action.

BxfMessage/@error: Acknowledgement messages shall include error(s) if the original message was rejected. Refer to examples above and the annex for additional information.

BxfMessage/@errorDescription: Acknowledgement messages should include an error description if the original message was rejected. Refer to examples above and the annex for additional information.

The acknowledgement shall precede the reply if the original message was a request.

6.5 Reply Messages

Per the BXF protocol documentation, applications that support asynchronous messaging in a connected environment must reply to all request messages that are received. Refer to remaining sections for example messages.

Conventions – Reply Messages

Replies shall only be generated in response to messages with a BxfMessage@messageType of "Request".

BxfMessage/@ext:usage: The usage is message specific. Refer to the reply sections of the document.

BxfMessage/@messageType: The message type shall be "Reply".
BxfMessage/@destination: Reply messages shall include a destination, which is the origin in the request message.

BxfMessage/@status: Reply messages shall include a status, which indicates if the request was processed successfully or not.

BxfMessage/@action: Reply messages should not contain actions.

BxfMessage/@error: Reply messages shall contain errors if the request was rejected.

BxfMessage/@errorDescription: Reply messages should include an error description if the original message was rejected.

6.6 Heartbeat Messages

Applications that support asynchronous messaging in a connected environment may use heartbeat messages to verify that another application is operational. Heartbeat capabilities are not required by the BXF protocol. It is important to note that regardless of whether an application is a sender of heartbeat messages, all applications are required by BXF (SMPTE 2021, Section 6.3.1) to acknowledge heartbeat messages.

6.6.1 Monitoring Other Applications

Applications may provide the capability to monitor other applications by creating and sending heartbeat messages. BXF describes additional capabilities that should be implemented with this functionality, such as a configurable interval for sending heartbeats. This functionality is not required by the BXF protocol.

Example messages:

Heartbeat
Heartbeat (targeted)

Conventions – Heartbeat Messages

BxfMessage/@ext:usage: The usage shall be "Application Heartbeat".

BxfMessage@messageType: The message type shall be "Heartbeat".

BxfMessage@destination: Heartbeat messages may be targeted to a single destination or sent to all destinations. The message originator is responsible for tracking application status by destination. Heartbeats with a destination attribute are targeted, while heartbeats with no destination are sent to all applications.

6.6.2 Acknowledging Heartbeat Messages

Applications shall provide the capability to acknowledge heartbeat messages. Heartbeat message acknowledgements are no different than any other type of acknowledgement in BXF.

Conventions – Heartbeat Acknowledgement Messages

BxfMessage/@ext:usage: The usage shall be "Application Acknowledgment".

BxfMessage@status: Heartbeat acknowledgements shall include a status of "OK". Error statuses are not applicable on heartbeat messages, since there is nothing to reject.

7 Transport Protocol (Normative)

7.1 Connection-Based Transport

The following sections describe connection oriented behaviors associated with BXF Messaging.

It is anticipated that prevalent technologies, such as web services, will be utilized within the transport layer. However, applications are free to utilize whatever technologies they desire within their API. Examples include, but are not limited to web services, HTTP Post, sockets and stored procedures.

- Web services should be used in situations requiring near real-time performance and where systems are loosely coupled, because they follow standardized protocols and procedures and can be language agnostic. They provide excellent performance and easily address accessibility issues including firewall access. Web services may not be appropriate for sending large messages, e.g. queries to retrieve all content metadata from another system.

- Sockets may also be used for situations requiring near real-time performance and where systems are loosely coupled and are situated on the same network. They may be used in other situations providing appropriate ports are exposed to connecting systems.

- Stored procedures may be used in cases involving tightly coupled integrations, i.e. in situations where the database is from the same vendor. Stored procedures provide extremely good performance, but require systems to be on the same network or need an intermediary to cross networks, as well as access to the server to communicate with an application database.

- File-based connections may be used for sending very large messages or in cases where near real-time performance is not a requirement. Either network file or FTP may be used as a delivery method. File-based connections may be useful for delivering very large queries or larger messages such as configuration values. Please note that as an alternative to file-based delivery for large files, message chunking may be used.

Applications have the flexibility to return an acknowledgement message as a return value or through a separate call to the server that the message originated from. The implementation approach depends on the type of application, the type of information and other factors, such as performance. Refer to Annex D, Protocol Transport, for additional information.

7.1.1 Default Ports

Per the BXF protocol, the default ports are 14544 and 14545. This does not preclude the use of other ports, such as port 80, for web services.

7.1.2 Loss of Communications

At any point in time, messages may be interrupted due to loss of connectivity. This loss may occur for any of a numbers of reasons, and may occur at any time.

As defined in the standard, Section 7.1.6 covers how to handle timeouts in communications. Anytime this situation occurs, the sending system can enter into a retry loop to reestablish communications. In addition, the receiving system will also enter into a retry loop.

During this time of loss of communications, no messaging can occur. Each affected system is responsible for buffering messages until such time as the systems are returned to normal communications.

7.1.3 Validating Sources and Destinations

Sources and Destinations shall be identified and configured prior to communication via BXF may occur. Once these names are configured, then systems may recognize messages from other systems.

There are two scenarios that can occur prior to proper configuration of these names, and are important to the integrity of operations between systems. These are:

- In the case of a message received from an unrecognized source, the receiving system may not make any reply. This will prevent external systems from “phishing” to attempt to breach security within the environment.

- In the case of a message received from a recognized source which incorrectly identifies the destination, such as the mismatch between the names of identified systems vs. the name as configured in the receiving system, the receiving system may NAK the message.

Systems that receive messages that do not conform to these two cases may choose to log these occurrences. This may be helpful when troubleshooting communications when configuring the systems initially.

7.1.4 Server-Only Applications

Server-only applications provide services for other applications to call, but do not have client functionality necessary to call services provided by other applications. Server-only applications shall acknowledge messages using the return value approach specified below. Server-only applications may be notified of changes to data (i.e., information messages) but cannot support a request for information since the corresponding reply message must be provided on a separate call to the originating application’s server.

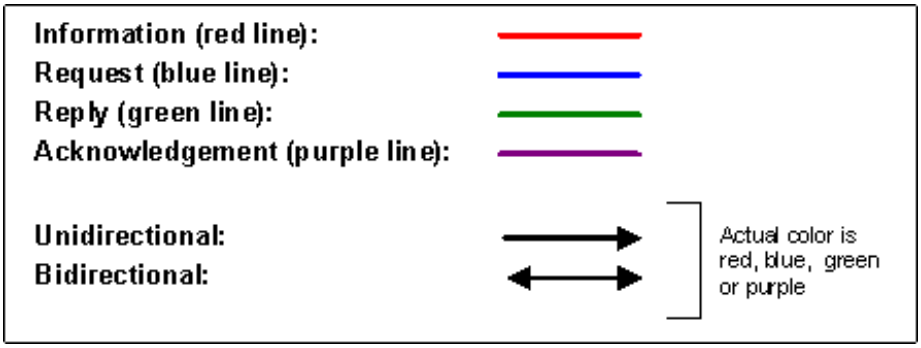
7.2 File-Based Transport

The basic messaging constructs used for connection-based transport are equally applicable to messaging using files. The only change that may be needed is the adjustment of the timeout configuration. This will allow implementations to use either a polling method or a notification method for processing of messages.

Note that there are cases where file-based operations will be necessary based on the size of returned messages. This implies that anytime connection based operations are implemented, file based may also be used.

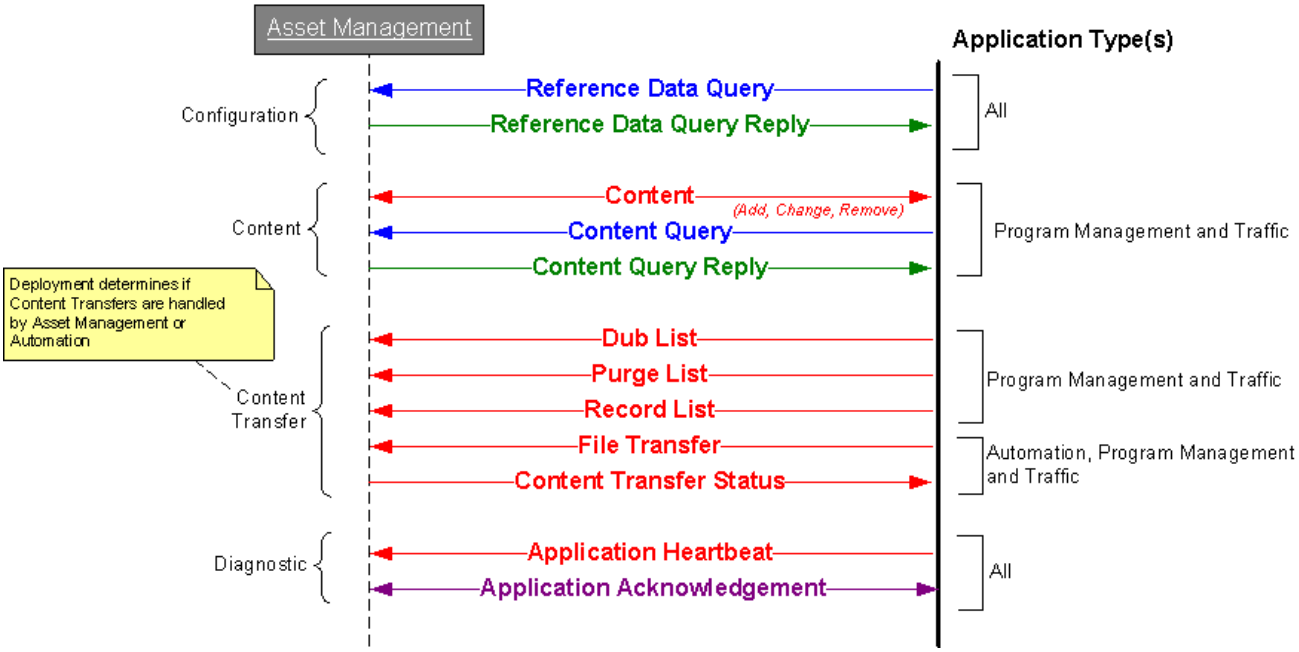
8 Message Type Usages by Application Type (Informative)

The following section identifies major communication links between applications based on their type and the type of information being sent (i.e., message type usage). Note that the message type usages in these sections are a subset of the possible usages defined in Annex B, Message Type Usages. Applications can implement as many usages as desired, based on their functionality.



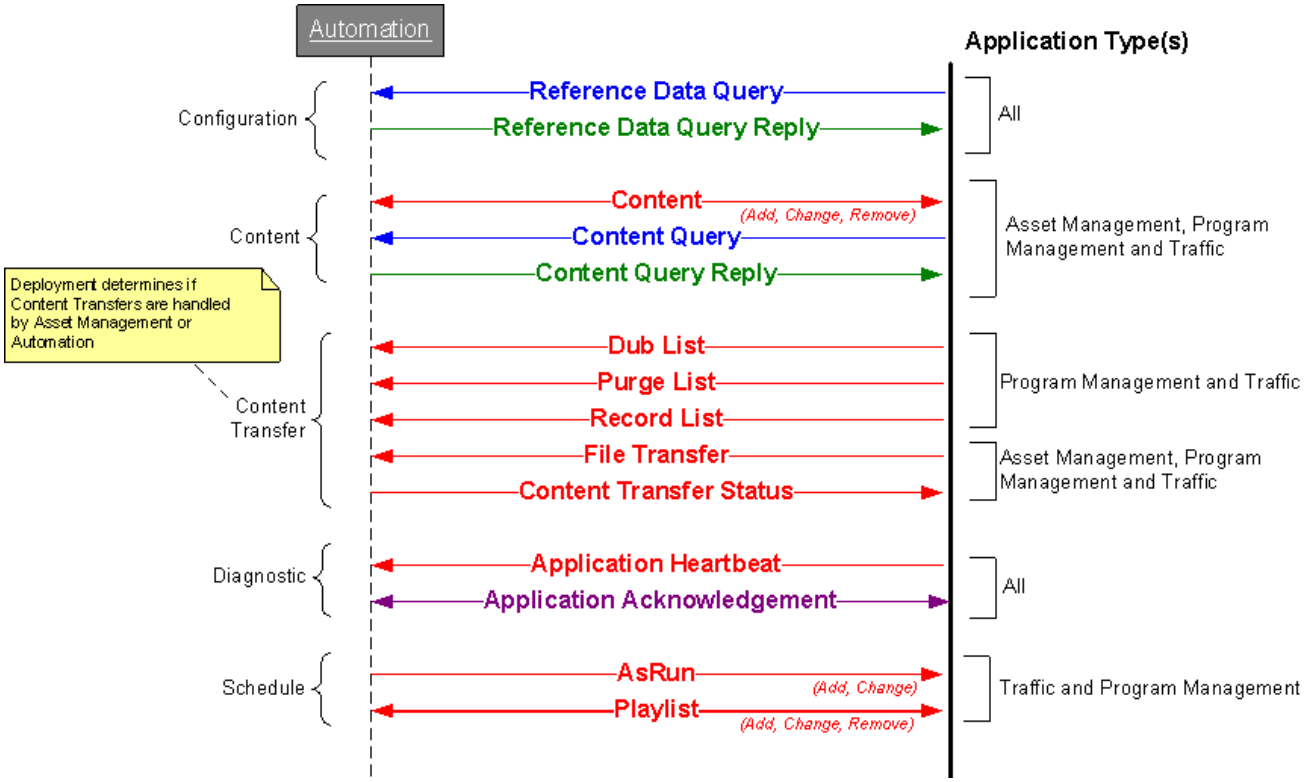
8.1 Asset Management

The following message usages are recommended for asset management applications.



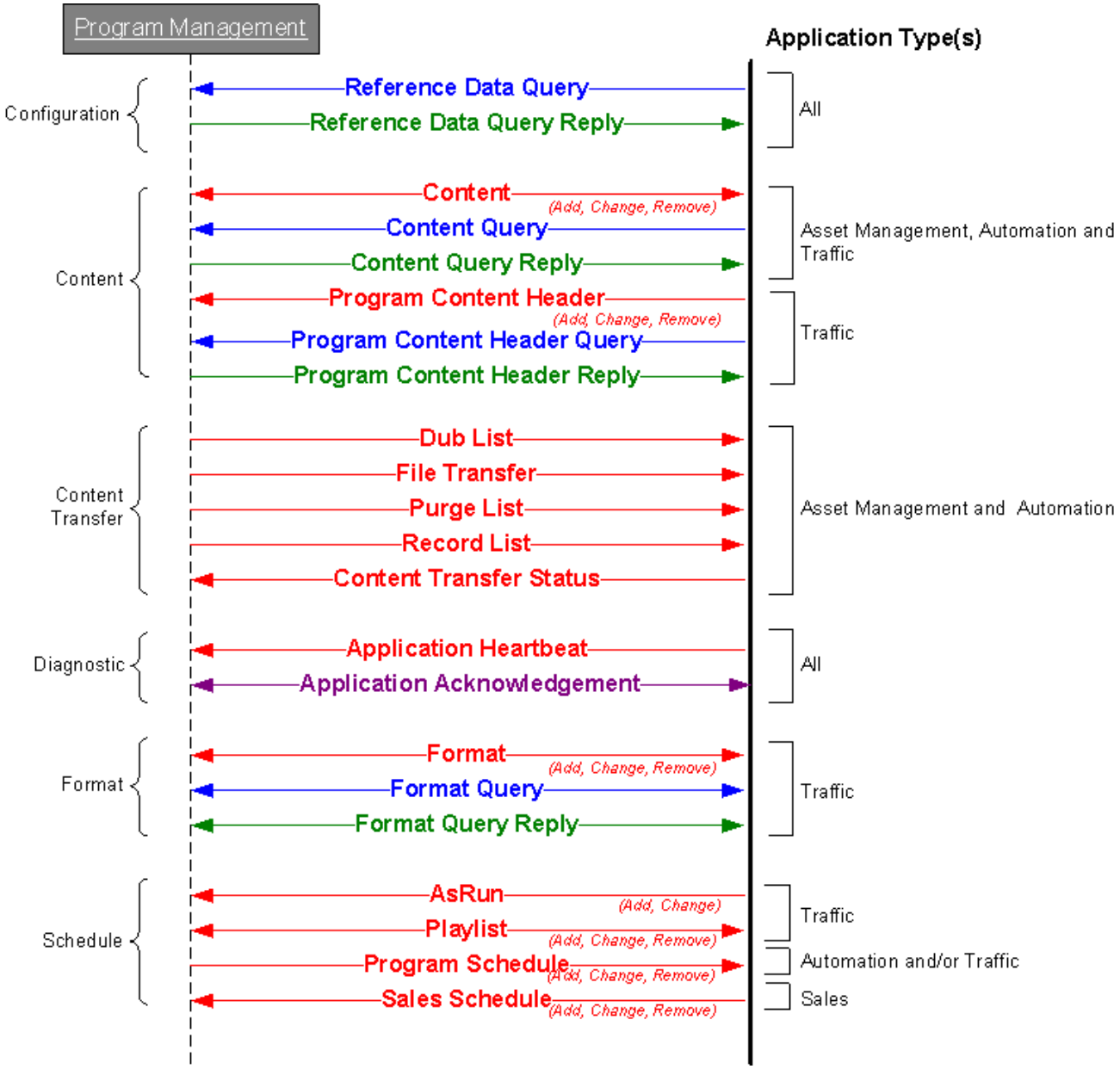
8.2 Automation

The following message usages are recommended for asset automation applications.



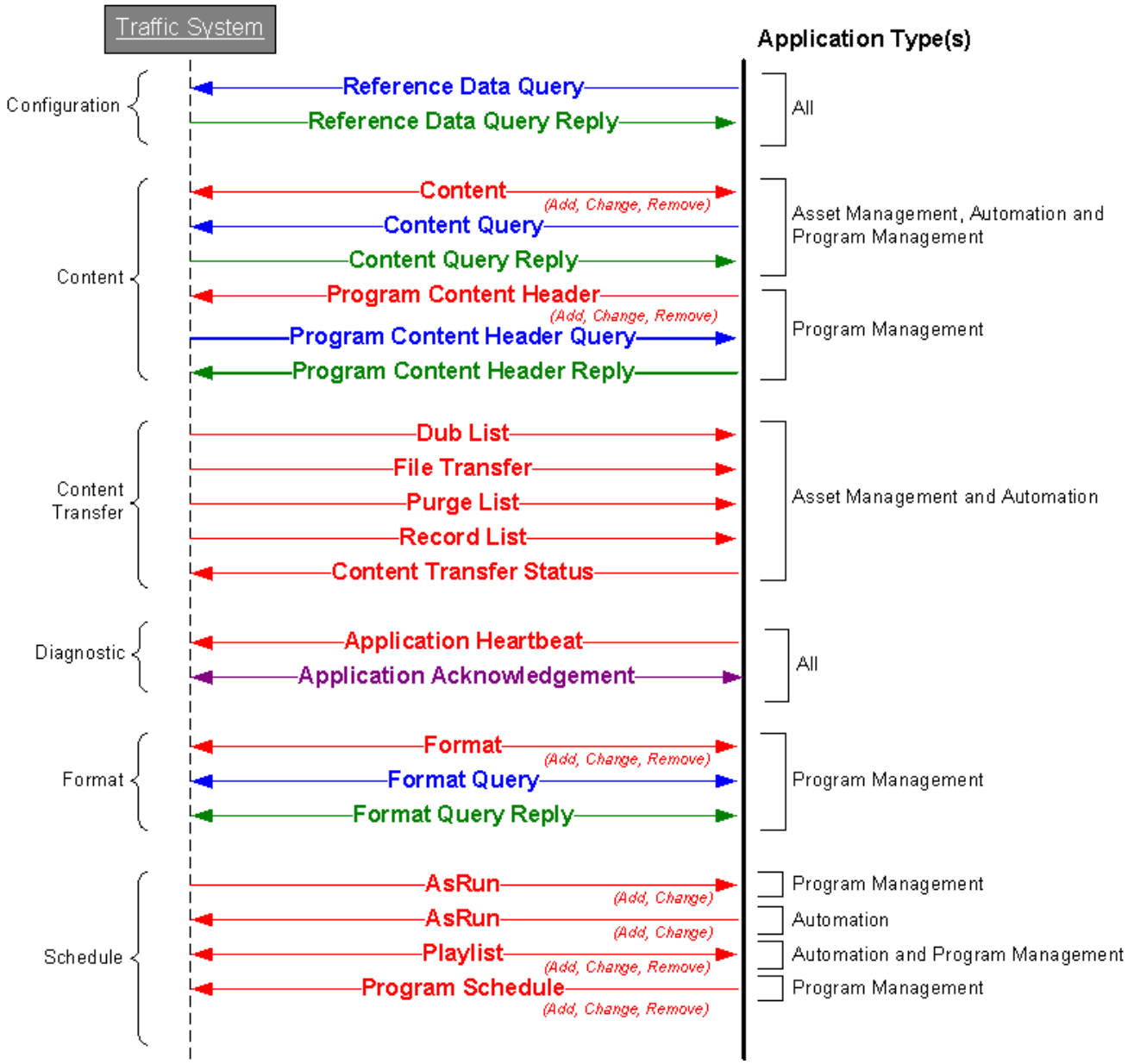
8.3 Program Management

The following message usages are recommended for program management applications.



8.4 Traffic

The following message usages are recommended for traffic applications.



9 Configuration

9.1 Overview (Informative)

The configuration elements within the BXF protocol allow applications to communicate configuration values that are commonly referenced in other types of messages (e.g., Playlist). Channel is a good example, since each application may have a different identifier for a channel, but the channel is in fact the same from the client's perspective. Enumerations can not be used for these values, since they are defined at runtime. Per Section "4 (Glossary/Configuration)" of the BXF documentation:

> *"There are several schema elements that are not specifically enumerated in the schema and must be configured between the two systems before the elements can be used."*

9.1.1 Configuration Processing Conventions (Normative)

It is important to note that the terms "reference data", "configuration", "configuration information" and "configuration values" all represent the same functionality in the following paragraphs and sections.

To optimally support configuration capabilities, one or more applications that participate in the BXF message exchange need to be designated as "sync points" for configuration information. These applications will provide mapping functionality to associate different application values. In addition, they may provide functionality for creating new configuration values and modifying existing values, so that the configuration information can be maintained over time. If the sync point concept is not utilized and instead each application provides its own mapping capabilities, then redundant mapping will be required within each application (this approach is not recommended).

Through the mapping process, sync point applications establish the canonical set of values that will be utilized by all participating applications. These values will be distributed by the sync point application to all participating applications. This means that participating applications will have their own (internal) configuration values and an associated sync point value. Refer to the example below for additional information.

The number of applications acting as sync points will vary based on the applications involved, the configuration types and the client. Each configuration type shall have only one owner (application). This is typically determined by the client, which identifies ownership of the configuration types.

If ownership of a configuration type is managed by an application with sync point functionality then the associated configuration values should be managed from the sync point.

9.1.2 Integrating without a Sync Point (Normative)

If the integrating applications do not have sync point functionality then configuration values may be mapped manually within each application. Configuration values are communicated between the applications in a non-programmatic manner and the applications are updated through existing functionality.

9.1.3 Integrating with a Sync Point (Normative)

9.1.3.1 Participating Application Responsibilities

Each application shall provide the following functionality to support configuration messages:

- Respond to a request for configuration types and values, providing support configuration types and values (if any)
- Process notification messages containing mappings between and application's values and the sync point's values

- Process notification messages containing updated or removed configuration values (optionally supported based on configuration type)

- Respond to a request to add configuration values (optionally supported based on configuration type)

These capabilities allow an application to provide its configuration values and maintain a 1:1 mapping between its values and the canonical values of the sync point. Applications are not required to maintain mappings for every application that they are connected to.

9.1.3.2 Sync Point Application Responsibilities

Sync point applications shall provide the following functionality to support configuration messages:

- Request the configuration values from other applications for one or more configuration types that are owned and process response messages containing each application's configuration values

- Provide a mapping facility (e.g., user interface, algorithms, etc.) to associate application values and identify the canonical set for a given configuration type

- Notify participating applications with the mappings between their values and the canonical values that will be used in BXF messages

- Notify participating applications whenever values are updated or removed

- Request configuration values be added and process response messages containing each application's associated configuration values

These capabilities allow a "sync point" application to identify and maintain the canonical set of values for BXF messaging. The "sync point" applications shall contain their own configuration values as well and these should be mapped just like any other applications values.

9.1.4 Configuration Example (Informative)

The following example shows values for three applications, where applications A and B are participating applications and application C is acting as the sync point for the configuration type of "Channel".

9.1.4.1 Initial Configuration

No mappings exist for the configuration data in applications A, B and C.

Application A			
ID	**Value**	**Mapped ID**	**Mapped Value**
0001	KAAA		
0002	KBBB		
0003	KCCC		

Application B			
ID	**Value**	**Mapped ID**	**Mapped Value**
2001	KA		
2002	KB		
2003	KC		

Application C (Sync Point)				
Canonical ID	**Canonical Value**	**Destination**	**Application ID**	**Application Value**
		Application C	9010	A
		Application C	9011	B
		Application C	9012	C

9.1.4.2 Request and Process Configuration Values

Application C requests configuration values from applications A and B and processes their responses, adding their configuration values to application C (resulting changes are highlighted).

Application A			
ID	Value	Mapped ID	Mapped Value
0001	KAAA		
0002	KBBB		
0003	KCCC		

Application B			
ID	Value	Mapped ID	Mapped Value
2001	KAAA		
2002	KB		
2003	KC		

Application C (Sync Point)				
Canonical ID	Canonical Value	Destination	Application ID	Application Value
		Application C	9010	A
		Application C	9011	B
		Application C	9012	C
		Application A	0001	KAAA
		Application A	0002	KBBB
		Application A	0003	KCCC
		Application B	2001	KAAA
		Application B	2002	KB
		Application B	2003	KC

9.1.4.3 Map Configuration Values

Application C maps the configuration values for the example configuration type "Channel", creating the canonical set of channel values (resulting changes are highlighted). Three unique channel values are identified: KAAA, KBBB and KCCC. Note that the mapping activity only occurs in application C. Note also that mapping of the values is one of the responsibilities of the sync point and is accomplished via the "mapping facilities" indicated in Section 9.1.3.

Application A			
ID	Value	Mapped ID	Mapped Value
0001	KAAA		
0002	KBBB		
0003	KCCC		

Application B			
ID	Value	Mapped ID	Mapped Value
2001	KAAA		
2002	KB		
2003	KC		

Application C (Sync Point)				
Canonical ID	Canonical Value	Destination	Application ID	Application Value
1	KAAA	Application C	9010	A
2	KBBB	Application C	9011	B
3	KCCC	Application C	9012	C
1	KAAA	Application A	0001	KAAA
2	KBBB	Application A	0002	KBBB
3	KCCC	Application A	0003	KCCC
1	KAAA	Application B	2001	KAAA
2	KBBB	Application B	2002	KB
3	KCCC	Application B	2003	KC

9.1.4.4 Send Mappings to Participating Applications

Application C sends the mapped values to applications A and B and their mappings are updated accordingly (resulting changes are highlighted). Applications A and B provide mapping functionality between their values and the canonical values, but they do not need to support mappings for all values in all applications.

Application A			
ID	**Value**	**Mapped ID**	**Mapped Value**
0001	KAAA	1	KAAA
0002	KBBB	2	KBBB
0003	KCCC	3	KCCC

Application B			
ID	**Value**	**Mapped ID**	**Mapped Value**
2001	KAAA	1	KAAA
2002	KB	2	KBBB
2003	KC	3	KCCC

Application C (Sync Point)				
Canonical ID	**Canonical Value**	**Destination**	**Application ID**	**Application Value**
1	KAAA	Application C	9010	A
2	KBBB	Application C	9011	B
3	KCCC	Application C	9012	C
1	KAAA	Application A	0001	KAAA
2	KBBB	Application A	0002	KBBB
3	KCCC	Application A	0003	KCCC
1	KAAA	Application B	2001	KAAA
2	KBBB	Application B	2002	KB
3	KCCC	Application B	2003	KC

9.1.4.5 Send Canonical Values on BXF Messages

When applications A, B and C send Format, Playlist, AsRun, Program Schedule, etc. messages, only the canonical values (i.e., KAAA, KBBB and KCCC) are sent for the channel identifier.

- Each application must translate its internal channel identifier into the canonical value when creating messages to send to other applications. This allows the sender to send the same message to all participating applications.

- Each application is responsible for translating the canonical channel identifier into its internal identifier when BXF messages are received and processed.

9.2 Recommended Conventions and Configuration Types (Normative)

Each application shall provide the capability to maintain a 1:1 mapping between its configuration values and the canonical values of the sync point.

The following configuration types are defined within the BXF protocol and are application and message type-specific.

9.2.1 BXF Configuration Types

The following configuration types are specified within the BXF protocol:

Configuration Type	Description
AgencyCode	Agency Code
Applicability	Content, Content Transfer and Schedule messages
Usage	**Optional** Applications that are agency code aware should provide their configuration values. Schedule-based agency codes are not recommended. Agency code is utilized in: • */Content/NonProgramContent/Details/Agency/AgencyCode • */ContentTransfer/NonProgramContent/Details/Agency/AgencyCode • */Schedule/ScheduledEvent/EventData/PrimaryEvent/NonProgramEvent/Details/Agency/AgencyCode • */Schedule/ScheduledEvent/EventData/NonPrimaryEvent/NonProgramEvent/Details/Agency/AgencyCode
AlternateIdType	Type of alternate identifier and the authoritative source of the content (e.g., ISCI, Custom)
Applicability	Content, Content Transfer and Schedule messages
Usage	**Recommended** Applications that utilize alternate identifiers for content should provide their configuration values. Alternate identifier type is utilized in: • */Content/ProgramContent/ContentMetadata/ContentId/AlternateId/@type • */Content/NonProgramContent/ContentMetadata/ContentId/AlternateId/@idType • */ContentTransfer/Content/ProgramContent/ContentMetadata/ContentId/AlternateId/@idType • */ContentTransfer/Content/NonProgramContent/ContentMetadata/ContentId/AlternateId/@idType • */Schedule/ScheduledEvent/Content/ContentId/AlternateId/@idType
AlternateIdSource	The source of the content that is using the alternative ID type of alternate identifier as the Authoritative Source of the content (e.g., traffic; automation; CDS; program management, ISCI, custom)
Applicability	Content, Content Transfer and Schedule messages
Usage	**Recommended** Applications that utilize alternate identifiers for content should identify them as the Authoritative Source configuration values. Authoritative source is utilized in: • */Content/ProgramContent/ContentMetadata/ContentId/AlternateId/@authoritativeSource • */Content/NonProgramContent/ContentMetadata/ContentId/AlternateId/@authoritativeSource • */ContentTransfer/Content/ProgramContent/ContentMetadata/ContentId/AlternateId/@authoritativeSource • */ContentTransfer/Content/NonProgramContent/ContentMetadata/ContentId/AlternateId/@authoritativeSource • */Schedule/ScheduledEvent/Content/ContentId/AlternateId/@authoritativeSource

AssetName	Tape storage format (e.g., BetaSp, DigiBeta, MiniDV)
Applicability	Content, Content Transfer and Schedule messages
Usage	**Recommended** Applications that are asset name aware should provide their configuration values. Asset name value is utilized in: • */Content/ProgramContent/ContentMetaData/Media/MediaLocation/Location/PhysicalAsset/@assetName • */Content/NonProgramContent/ContentMetadata/Media/MediaLocation/Location/PhysicalAsset/@assetName • */ContentTransfer/Content/ProgramContent/ContentMetadata/Media/MediaLocation/Location/PhysicalAsset/@assetName • */ContentTransfer/Content/NonProgramContent/ContentMetadata/Media/MediaLocation/Location/PhysicalAsset/@assetName • */Schedule/ScheduledEvent/Content/Media/MediaLocation/Location/PhysicalAsset/@assetName
Classification	Program or spot classifications that may restrict the movement of a program or spot from one event to another by the operator
Applicability	Schedule messages
Usage	**Optional** Applications that are classification aware may provide their configuration values. Classification is utilized in: • */Schedule/ScheduledEvent/EventData/PrimaryEvent/ProgramEvent/Constraints/Rules/Classification • */Schedule/ScheduledEvent/EventData/PrimaryEvent/NonProgramEvent/Constraints/Rules/Classification • */Schedule/ScheduledEvent/EventData/NonPrimaryEvent/NonProgramEvents/Constraints/Rules/Classification
ContentType	Content Type (e.g., network, local, entertainment, news)
Applicability	Schedule messages
Usage	**Optional** Applications that are content type aware may provide their configuration values. Content type is utilized in • */Schedule/ScheduledEvent/ContentType • */Schedule/AsRun/CompleteAsRun/ContentType
FederalSource	Federal Source for the event (e.g., live, recorded or network)
Applicability	Schedule messages
Usage	**Recommended** Applications that are federal source aware should provide their configuration values. Federal source is utilized in: • */Schedule/ScheduledEvent/EventData/FederalSource
MacroName	Macro Name
Applicability	Content, Content Transfer, Format and Schedule messages

Usage	**Optional** Applications that are macro aware may provide their configuration values. Macro name is utilized in: • */Content/ProgramContent/ContentMetaData/Media/ProfileMacro/MacroName • */Content/NonProgramContent/ContentMetadata/Media/ProfileMacro/MacroName • */ContentTransfer/Content/ProgramContent/ContentMetadata/Media/ProfileMacro/MacroName • */ContentTransfer/Content/NonProgramContent/ContentMetadata/Media/ProfileMacro/MacroName • */Format/FormatStructure/FormatElements/NonPrimaryElements/Macros • */Schedule/ScheduledEvent/EventData/MacroEvent/MacroName • */Schedule/ScheduledEvent/EventData/NonPrimaryEvent/Macros/MacroName
MessageDestination	Intended destination of a BXF message
Applicability	All messages
Usage	**Optional** The destination values are configured during system setup. Destination values are not mapped so they are not a recommended Configuration type. Destination is utilized in: • BxfMessage/@destination
MessageOrigin	The origin of a BXF message
Applicability	All messages
Usage	**Optional** The origin values are configured during system setup. Origin values are not mapped so they are not a recommended configuration type. Origin is utilized in: • BxfMessage/@origin
NonPrimaryEventName	Non-primary event type (e.g., key, GPI, tone, logo, bug, crawl)
Applicability	Schedule messages
Usage	**Recommended** Applications that are non-primary event name aware should provide their configuration values. Non-primary event name is utilized in: • */Schedule/ScheduledEvent/EventData/NonPrimaryEvent/NonPrimaryEventName
ProductCode	Product Code that categorizes a product (e.g., fast food, auto, beverages)
Applicability	Content, Content Transfer and Schedule messages
Usage	**Optional** Applications that are product code aware may provide their configuration values. Product code is utilized in: • */Content/NonProgramContent/Details/Product/ProductCode • */ContentTransfer/Content/NonProgramContent/Details/Product/ProductCode • */Schedule/ScheduledEvent/EventData/PrimaryEvent/NonProgramEvent/Detai

	ls/Product/ProductCode • */Schedule/ScheduledEvent/EventData/NonPrimaryEvent/NonProgramEvent/Details/Product/ProductCode
ProtectionSourceName	Name of the type of protection source to be used (e.g., ProtSrc1)
Applicability	Schedule messages
Usage	**Recommended** Applications that are protection source aware should provide their configuration values. Protection source is utilized in: • */Schedule/ScheduledEvent/EventData/Protection/ProtectionSourceName
RouterSourceName	Router Source Name (e.g., Router1, Srv2)
Applicability	Content, Content Transfer and Schedule messages
Usage	**Recommended** Applications that are router source aware should provide their configuration values. Router source is utilized in: • */Content/ProgramContent/ContentMetaData/Media/MediaLocation/Location/RouterSource/Name • */Content/NonProgramContent/ContentMetadata/Media/MediaLocation/Location/RouterSource/Name • */ContentTransfer/Content/ProgramContent/ContentMetadata/Media/MediaLocation/Location/RouterSource/Name • */ContentTransfer/Content/NonProgramContent/ContentMetadata/Media/MediaLocation/Location/RouterSource/Name • */Schedule/ScheduledEvent/Content/Media/MediaLocation/Location/RouterSource/Name
SpotType (NonProgramDetailType)	Spot Type for non-program content (e.g., barter spot, promo, PSA, ID, comments)
Applicability	Content, Content Transfer and Schedule messages
Usage	**Recommended** Applications that are spot type aware should provide their configuration values. Spot type is utilized in: • */Content/NonProgramContent/Details/SpotType • */ContentTransfer/Content/NonProgramContent/Details/SpotType • */Schedule/ScheduledEvent/EventData/PrimaryEvent/NonProgramEvent/Details/SpotType • */Schedule/ScheduledEvent/EventData/NonPrimaryEvent/NonProgramEvents/Details/SpotType
SpotSalesClassification	Spot rate section for non-program content (e.g., P1, R2, NP, comments)
Applicability	Schedule messages
Usage	**Optional** Applications that are spot sales classification (i.e., “rate section”) aware may provide their configuration values. Spot sales classification is utilized in:

	• */Schedule/ScheduledEvent/EventData/PrimaryEvent/NonProgramEvent/SpotSalesClassification • */Schedule/ScheduledEvent/EventData/NonPrimaryEvent/NonProgramEvents/SpotSalesClassification
TransitionPattern	Available patterns for transitions from one video source to another (Fade, Wipe)
Applicability	Schedule messages
Usage	**Recommended** Applications that are video transition pattern aware should provide their configuration values. Transition pattern is utilized in: • */Schedule/ScheduledEvent/EventData/Transitions/VideoTransitions/TransitionPattern

9.2.2 Recommended Extension Types

The following configuration types are recommended for the BXF protocol:

Condition	Result
AspectRatio	Identifies the aspect ratio for the video (e.g., 16:9, 4:3)
Applicability	Content, Content Transfers, Schedules
Usage	**Recommended** Applications that are Aspect Ratio aware should provide their Aspect Ratio configuration values. BXF does not provide a configurable element for aspect ratio, so these values should be provided in the following location: • */Content/ProgramContent/ContentMetaData/Media/PrivateInformation/AspectRatio • */Content/NonProgramContent/ContentMetadata/Media/PrivateInformation/AspectRatio */ContentTransfer/Content/ProgramContent/ContentMetadata/Media/PrivateInformation/AspectRatio • */ContentTransfer/Content/NonProgramContent/ContentMetadata/Media/PrivateInformation/AspectRatio • */Schedule/ScheduledEvent/Content/Media/PrivateInformation/AspectRatio
BreakCode	Codes to allow systems to configure breaks for certain types of spots (e.g. Promo Only, Barter, etc.)
Applicability	Format
Usage	**Recommended** Applications that are BreakCode aware should provide their Break Code configuration values. BreakCode value is utilized within: • BxfMessage/BxfData/Format/FormatStructure/FormatElements/PrivateInformation/BreakCode
Channel	A unique identifier for a channel

Applicability	Content, Content Transfers, Schedules
Usage	**Recommended** Applications that are Channel aware should provide their Channel configuration values. Channel value is utilized within: • BxfMessage/BxfData/Schedule/Channel/Name element on schedule messages • BxfMessage/BxfData/Content/ProgramContent/ContentMetaData/Policy/AssignedChannels on program content messages • BxfMessage/BxfData/Content/NonProgramContent/ContentMetaData/Policy/AssignedChannels on non-program content messages
Distributor	Distributor of the content; related to the contract data (e.g., Sony, Warner)
Applicability	Content, Program Contracts
Usage	**Recommended** Applications that are Distributor aware for content should provide their Distributor configuration values. Distributor value is utilized within: • */Content/ProgramContent/Contract/Distributor • */Content/ProgramContract/Distributor
Genre	Type (or grouping) of content based on the storyline or style of the content. (e.g., police drama, western, game show, children's show)
Applicability	Content, Content Transfers, Schedules
Usage	**Recommended** Applications that are Content aware should provide their Genre configuration values. Genre value is utilized within: • */ContentMetaData/Genre
ParentalRatingRegion	Parental Rating Region for the content (e.g., 01 – United States, 02 – Canada (*PMCP Values are used))
Applicability	Content, Formats (Private Information)
Usage	**Recommended** Applications that are Parental Rating Region aware for content should provide their Parental Rating Region configuration values. Parental Rating Region value is utilized within: • */Content/ProgramContent/ParentalRating/@region • */Format/PrivateInformation/@region
ParentalRatingDimension	Parental Rating Dimension for the content (e.g., TV Parental)
Applicability	Content, Format (Private Information)
Usage	**Recommended** Applications that are Parental Rating Dimension aware for content should provide their Parental Rating Dimension configuration values. Parental Rating Dimension value is utilized within: • */Content/ProgramContent/ParentalRating/@dimension

	• */Format/PrivateInformation/@dimension
ParentalRatingValue	Parental Rating Value for the content (e.g., TV-G, TV-PG)
Applicability	Content, Format (Private Information)
Usage	**Recommended** Applications that are Parental Rating Value aware for content should provide their Parental Rating Value configuration values. Parental Rating Region value is utilized within: • */Content/ProgramContent/Parental Rating/@value • */Format/PrivateInformation/@value
QualityRatingRegion ***(Quality = number of stars can use .5 for half stars.)***	Quality Rating Region for the content (e.g., 01 – United States, 02 – Canada (*PMCP Values are used))
Applicability	Content
Usage	**Recommended** Applications that are Quality Rating Region aware for content should provide their Quality Rating Region configuration values. Parental Rating Region value is utilized within: • */Content/ProgramContent/QualityRating/@RatingRegion
QualityRatingDimension	Quality Rating Dimension for the content (e.g., Chicago Tribune)
Applicability	Content
Usage	**Recommended** Applications that are Quality Rating Dimension aware for content should provide their Quality Rating Dimension configuration values. Quality Rating Dimension value is utilized within: • */Content/ProgramContent/QualityRating/@Dimension
RightsType	The type of program rights; related to the contract data (e.g., view-based, traditional broadcast, box office)
Applicability	Program Contracts
Usage	**Recommended** Applications that are Rights Type aware for content should provide their Rights Type configuration values. . Rights Type value is utilized within: • */Content/ProgramContract/Rights/RightsType

9.3 Requesting Configuration Information (Queries) (Normative)

The following sections identify how an application can request the configuration of another application. Each section is annotated, indicating if the functionality is required, optional or to be avoided by applications.

9.3.1 Obtaining All Configuration Values

The conventions for handling BXF queries for configuration information are described in the following sections. It is strongly recommended that query capabilities be supported.

9.3.1.1 Request Message

Applications may provide the capability to consume a request for all of their configuration values. It is strongly recommended.

Applications may provide the capability to create a request for another application's configuration values, as well as the functionality to consume the corresponding reply.

Example message for requesting all configuration values:

Configuration – Get All

Recommended Conventions – Configuration Query Request

BxfMessage/@ext:usage: The usage shall be "Reference Data Query".

BxfMessage/@messageType: The message type shall be "Request".

BxfQuery/WhereClause: For Configuration Query Request, the <WhereClause> element for "Configuration/Name="*"" shall be supported, which indicates that all configuration types and values are being requested.

9.3.1.2 Reply Message

Applications may provide the capability to reply to a configuration query request. It is strongly recommended.

Example message for replying to a configuration value request:

Configuration – Get All Reply

Recommended Conventions – Configuration Query Reply

An application should return all of the configuration types that it supports.

BxfMessage/@ext:usage: The usage shall be "Reference Data Query Reply".

BxfMessage/@messageType: The message type shall be "Reply".

BxfQueryResponse/Configuration/Name: The type of configuration information (e.g., Genre). If no values are defined for a type, then the type shall be returned without values (as indicated in the example message).

BxfQueryResponse/Configuration/ConfigurationData/@ext:id: The "id" extension attribute is provided on each ConfigurationData element. It shall contain the application's unique identifier for the configuration value.

BxfQueryResponse/Configuration/ConfigurationData/@ext:description: The "description" extension attribute is provided on each ConfigurationData element. It may be used to describe a value that is not intuitive.

BxfQueryResponse/Configuration/ConfigurationData/@ext:supportedActions: The "supportedActions" extension attribute is provided on each Configuration Data element and should indicate the actions that the application supports for this configuration type (e.g., genre value can be added, updated and removed, while channel identifiers cannot be modified in any way). If this attribute is not provided then the supported actions should be defaulted to none.

Additional conventions are specified in **Reply Messages**.

9.3.2 Obtaining Type- and Value-Specific Information

The conventions for handling BXF queries for type- and value-specific configuration information are described in the following sections.

9.3.2.1 Request Message

Applications may provide the capability to consume a request for type- and value-specific information.

Applications may provide the capability to create a request for another application's type- and value-specific information, as well as the functionality to consume the corresponding reply.

Example messages for requesting type- and value-specific information:

Configuration – Get Type Specific Data
Configuration – Get a Specific Value by Id
Configuration – Get a Specific Value by Value

Recommended Conventions – Type-Specific Configuration Query Request

BxfMessage/@ext:usage: The usage shall be "Reference Data Query".

BxfMessage/@messageType: The message type shall be "Request".

BxfQuery/WhereClause: For Type-Specific Configuration Query Request, only one type may be specified in the <WhereClause> element. One configuration data value (or id) may also be included. Examples:

- By type only: Configuration/Name="SpotType"
- By type and id: Configuration/Name="SpotType and Configuration/ConfigurationData@ext:id="12333""
- By type and value: Configuration/Name="SpotType and Configuration/ConfigurationData="Promo""

Advanced queries that include complex conditional logic and/or operators other than equality (e.g., >, =, <) are not recommended.

9.3.2.2 Reply Message

Applications shall provide the capability to reply to a type-specific configuration query request.

Example message:

Configuration – Get Type Specific Reply
Configuration – Get Value Specific Reply

Recommended Conventions – Type-Specific Configuration Query Reply

BxfMessage/@ext:usage: The usage shall be "Reference Data Query Reply".

BxfMessage/@messageType: The message type shall be "Reply".

BxfQueryResponse/Configuration/Name: The type of configuration information (e.g., Genre). If no values are defined for the requested type then the type shall be returned without values.

BxfQueryResponse/Configuration/ConfigurationData/@ext:id: An "id" extension attribute shall be provided on each ConfigurationData element. It contains the application's unique identifier for the configuration value.

BxfQueryResponse/Configuration/ConfigurationData/@ext:supportedActions: A "supportedActions" extension attribute shall be provided on each Configuration Data element indicating the actions that the application supports for this configuration type (e.g., genre value can be added, updated and removed, while channel identifiers cannot be modified in any way).

Additional conventions are specified in **Reply Messages**.

9.3.3 Using Advanced Queries

The BXF protocol supports advanced querying capabilities and includes conditional logic (e.g., and, or logic) and operators (e.g., <, >, =, etc.).

In the context of configuration, these capabilities should be avoided since they will require significant effort to develop.

9.4 Requesting New Configuration Values be Added (Normative)

The following sections identify how an application shall request that new configuration values be added to other applications. All of these capabilities are viewed as important to the implementation.

9.4.1 Request Message

Applications should provide the capability to consume a configuration request message that contains new configuration values. If the request contains new values for a configuration type that the application does not support, then the message should be rejected.

Applications may provide the capability to create a configuration request message that contains new configuration values. The application that is acting as the "sync point" shall have this capability.

Example message containing a request to add configuration values:

Configuration - Add

Recommended Conventions – Configuration Request

BxfMessage/@ext:usage: The usage shall be "Reference Data Request".

BxfMessage@messageType: The message type shall be "Request".

Configuration/@action: The message shall contain at least one new configuration value (i.e., "add" action) to be considered a request.

Configuration/@action: For Configuration Request, the message may also contain other configuration changes (i.e., "update" and/or "remove" actions). The corresponding reply will only reference the configuration values that were added.

Configuration/ConfigurationData/@ext:originId: An "originId" extension attribute shall be provided on each ConfigurationData element. It contains the originating application's unique identifier.

Configuration/ConfigurationData/@ext:originValue: An "originValue" extension attribute may be provided on each ConfigurationData element. It contains the originating application's value. This attribute shall be included if the originating application's value is different than the value it is requesting the destination application add.

9.4.2 Reply Message

Applications shall provide the capability to reply to a configuration request message.

Reply example:

Configuration – Add Reply

Recommended Conventions – Configuration Reply

BxfMessage/@ext:usage: The usage shall be “Reference Data Reply”.

BxfMessage/@messageType: The message type shall be “Reply”.

BxfQueryResponse/Configuration/ Name: The type of configuration information (e.g., Genre).

Configuration/ConfigurationData/@ext:id: An “id” extension attribute shall be provided on each ConfigurationData element. It contains the application’s unique identifier for the configuration value.

Configuration/ConfigurationData/@ext:originId: An “originId” extension attribute shall be provided on each ConfigurationData element. It contains the originating application’s unique identifier (i.e., the application that requested the change in the first place).

Configuration/@action: Action values should not be included on reply messages.

Additional conventions are specified in **Reply Messages**.

9.5 Notifying Others when Values or Mappings Change (Normative)

The following sections identify how an application shall notify other applications that its configuration values have changed and/or that destination application values need to change. All of these capabilities are considered to be important to the implementation.

9.5.1 Information Message – Configuration Update Values

Applications shall provide the capability to consume an information message that contains updates to their configuration values or the values of the originator.

Applications may provide the capability to create configuration messages that contain updated values. The application that is acting as the “sync point” shall have this capability.

Example message containing a notification to update configuration values:

Configuration – Update Values
Configuration – Update and Remove Values

Recommended Conventions – Configuration Update Values

BxfMessage/@ext:usage: The usage shall be “Reference Data”.

BxfMessage/@messageType: The message type shall be “Information”.

BxfMessage/BxfData/@action: The action shall be “update”. The sub-elements in the message may contain “remove” actions, but should not contain “add” actions, since information messages do not have corresponding reply messages.

BxfMessage/BxfData/Configuration/@ext:updateType: The "updateType" extension attribute shall be provided on the ConfigurationData element and its value is "Values", to indicate value-related changes.

Elements with identifiers and values shall be processed as updates:

BxfMessage/BxfData/Configuration/ConfigurationData /@ext:id: An "id" extension attribute shall be provided on each ConfigurationData element that contains an updated application value.

BxfMessage/BxfData/Configuration/ConfigurationData /@ext:originId: An "originId" extension attribute shall be provided on each ConfigurationData element that contains an updated originating application value.

Elements with identifiers, but no values, shall be ignored:

BxfMessage/BxfData/Configuration/ConfigurationData /@ext:id: If an "id" extension attribute is provided with no corresponding value, then the update shall be ignored for this ConfigurationData element and the corresponding acknowledgement message shall contain a warning.

BxfMessage/BxfData/Configuration/ConfigurationData /@ext:originId: If an "originId" extension attribute is provided with no corresponding value, then the update shall be ignored for this ConfigurationData element and the corresponding acknowledgement message shall contain a warning.

Elements with no identifiers shall be informational:

BxfMessage/BxfData/Configuration/ConfigurationData /@ext:id: If a ConfigurationData element value is provided with no corresponding "id" extension attribute, then the value is being provided for informational purposes. No update shall occur.

BxfMessage/BxfData/Configuration/@ext:originId: If an "originValue" extension attribute exists with no corresponding "id" extension attribute, then the value is being provided for informational purposes. No update shall occur.

9.5.2 Information Message – Configuration Update Mappings

Applications shall provide the capability to consume information messages that contain updates to their configuration value mappings (i.e., the mapping between their values to the originator values).

Applications may provide the capability to create configuration messages that contain updated mappings. The application that is acting as the "sync point" shall have this capability.

Example message containing a notification to update configuration mappings:

Configuration – Update Mappings

Recommended Conventions – Configuration Update Mappings

BxfMessage/@ext:usage: The usage shall be "Reference Data".

BxfMessage/@messageType: The message type shall be "Information".

BxfMessage/BxfData/@action: The action shall be "update". The sub-elements in the message may contain "remove" actions but not "add" actions, since information messages do not have corresponding reply messages.

BxfMessage/BxfData/Configuration/@ext:updateType: The "updateType" extension attribute shall be provided on the Configuration element and its value be shall be set to "Mappings" since this is a mapping-related change.

BxfMessage/BxfData/Configuration/ConfigurationData/@ext:id: An “id” extension attribute shall be provided on each ConfigurationData element. It contains the application’s unique identifier. It is used to identify the application value that is being mapped or unmapped.

BxfMessage/BxfData/Configuration/ConfigurationData/@ext:originId: An “originId” extension attribute shall be provided on each ConfigurationData element. It contains the originating application’s unique identifier (i.e., the value to map). If a value is being unmapped, then the “originId” value should contain an empty string (e.g., “”).

BxfMessage/BxfData/Configuration/ConfigurationData/@ext:originValue: An “originValue” extension attribute shall be provided on each ConfigurationData element. It contains the originating application’s value. It is provided for informational purposes.

BxfMessage/BxfData/Configuration/ConfigurationData/@ext:value: The ConfigurationData element may contain a value but it is not required. It may be provided for informational purposes.

9.5.3 Information Message – Remove Values

Applications shall provide the capability to consume information messages that contain configuration value removals.

Applications may provide the capability to create information messages containing obsolete configuration values that are no longer in use (i.e., have been removed). The application that is acting as the “sync point” shall have this capability.

Note: An application accepting a Remove message may choose to deactive the value in their system if maintaining the value in their system is required.

Example message containing a notification to remove a configuration value:

Configuration – Remove Values

Recommended Conventions – Configuration Remove Values

BxfMessage/@ext:usage: The usage shall be “Reference Data”.

BxfMessage/@messageType: The message type shall be “Information”.

BxfMessage/BxfData/@action: The action shall be “remove”.

BxfMessage/BxfData/Configuration/ConfigurationData/@ext:id: An “id” extension attribute shall be provided so the item can be removed.

Note: The origin’s mapping and value should be removed from the destination application.

BxfMessage/BxfData/Configuration/ConfigurationData/@ext:value: The value shall be provided.

Removal may not actually remove the value from the application, but disable it or change the status to “Marked for Deletion”. The key requirement is that an application shall not send other message types using configuration values that were removed.

10 Content

10.1 Overview (Informative)

The Content element within the BXF protocol allows applications to share the logical and physical aspects of content. BXF definition for content metadata:

> *"Information concerning a specific piece of content t hat further describes that content including synopsis, video, audio and captioning, duration, talent, year produced, producer, etc."*

10.1.1 Program Content Conventions and Levels (Normative)

The content-related messages support the following capabilities:

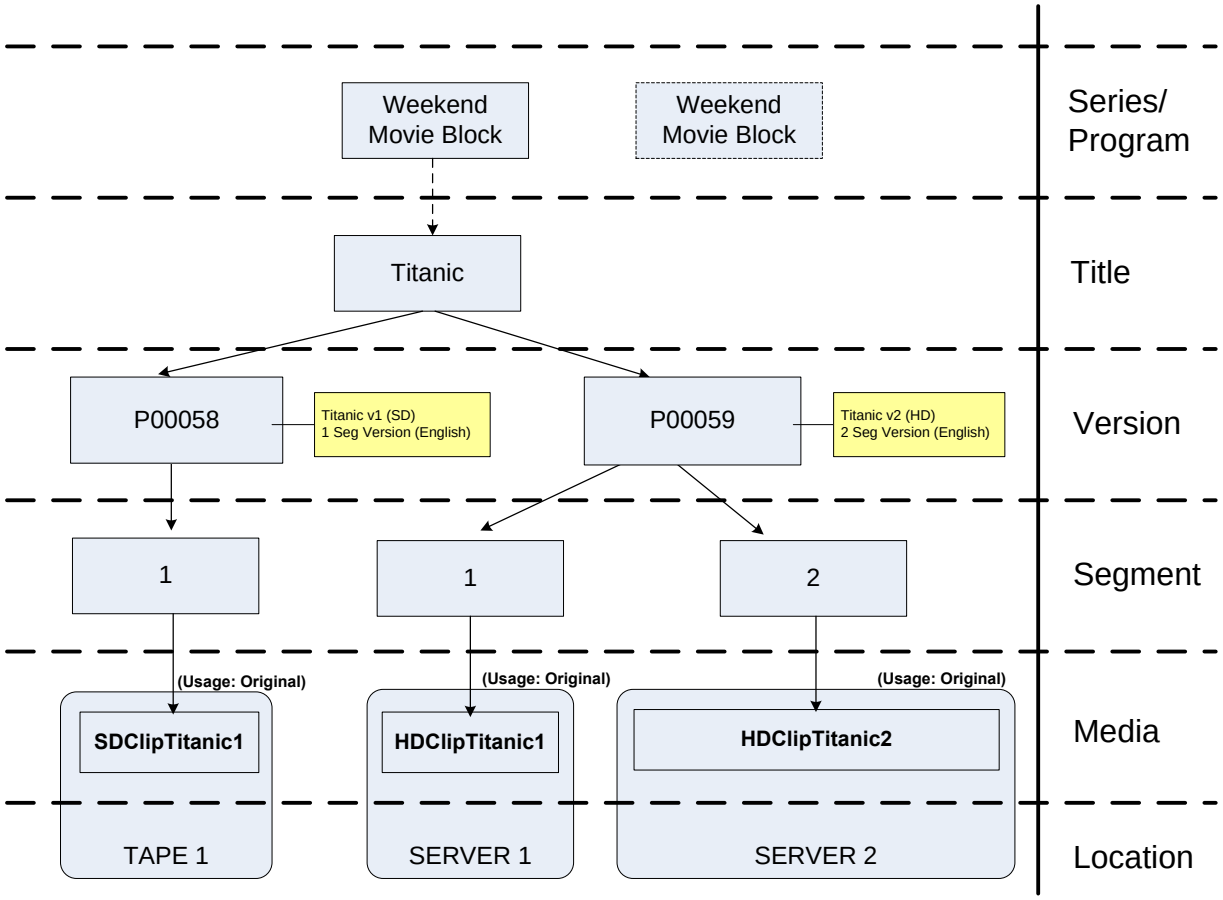
- The capability to generate and consume notification messages relating to content additions, updates and removals. These messages can originate from any participating application.
- The capability to reply to request messages for content-related information including "all" content and/or targeted content information that exists within an application.
- The capability to request content-related information.

Applications provide different approaches for uniquely identifying content based on the type of content (i.e., program, non-program) and other factors. The recommended approach to content messaging utilizes the alternate identifier capabilities of the BXF protocol to uniquely identify content-related elements using global identifiers, or GUIDs. The GUIDs may be provided in addition to other content identifiers (e.g., house number, ISCI). It is recommended that each application associate the GUID values with their own corresponding content information.

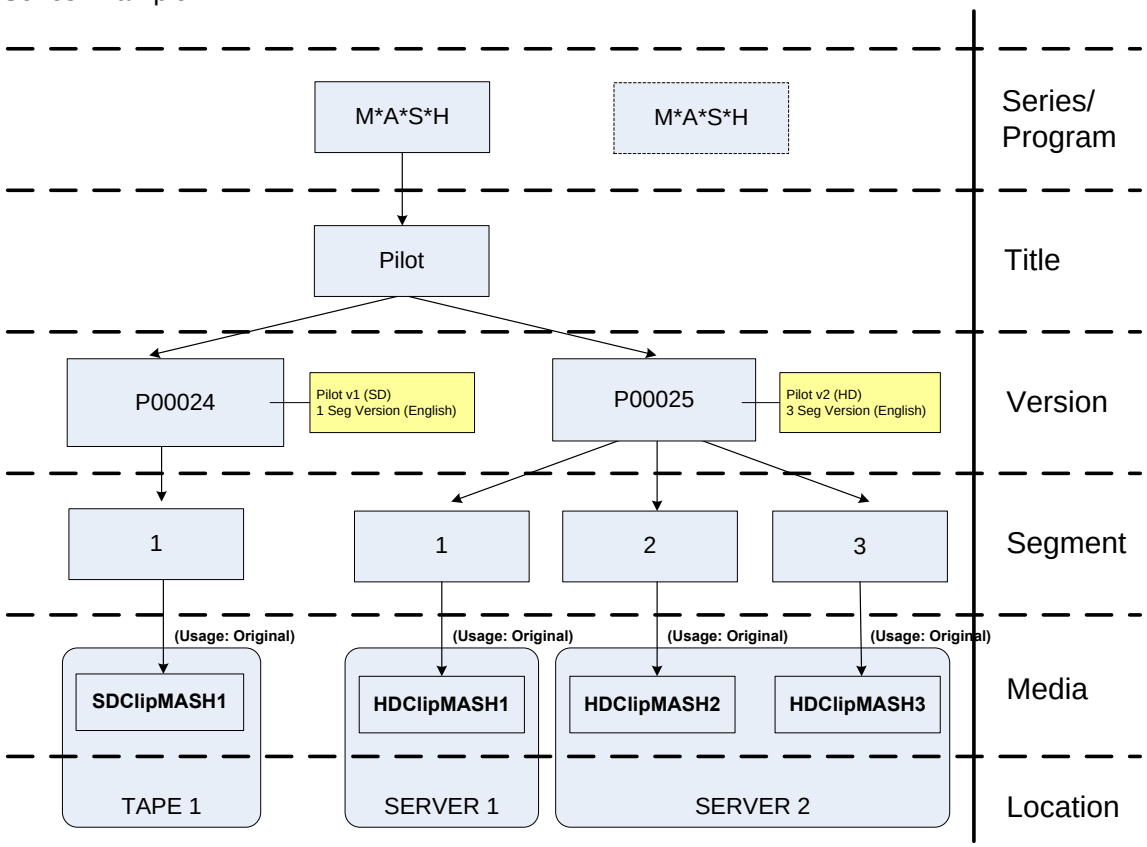
It is anticipated that content-related GUIDs will not be visible within applications by users and will primarily be used to uniquely identify content in a standardized manner with other applications. One or more visible content identifiers such as house number, ISCI, ISAN, etc., are provided on all content elements within a message as described in the following sections.

The following diagram identifies the different content levels for a movie (e.g., Titanic) followed by a series example and one that involves only a news program. Regardless of whether it is a single program (movie) or a series, the same structure would be used; however, a series grouping exists when the program belongs to a series. These examples also support transactions that describe a limited program only grid used during the planning stages of a schedule and which support EPG metadata, but do not yet have the details for a complete playlist. Refer to the definitions in the next section for additional information.

Single Program (Movie) Example:



Series Example:



New Program Example:

10.1.1.1 Program Content Structure (Normative)

The following sections describe the different program content levels and specify how each level is identified within BXF.

10.1.1.1.1 Programs and Series

Both Series' and Programs are used to represent a collection of programs and appear at the same content level. Programs are content without a continuing story line, such as news programs, public affairs programs, paid programs and movie blocks. A series is program content that is produced as a continuing storyline and normally has many of the same characters or features the same content (e.g., M*A*S*H).

Program and series are mutually exclusive and they shall not be applied to the same program content.

10.1.1.1.1.1 Program

***/Content/@version:** The version attribute shall be "Program".

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@idType:** The idType attribute shall be "ProgramId". This represents the technical key for the program.

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@authorativeSource:** The authorative source shall be "GlobalId". The global identifier is a common source that is shared by all applications.

10.1.1.1.1.2 Series

***/Content/@version:** The version attribute shall be "Series".

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@idType:** The idType attribute shall be "SeriesId". This represents the technical key for the series.

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@authorativeSource:** The authorative source shall be "GlobalId". The global identifier is a common source that is shared by all applications.

10.1.1.1.2 Title

A title is program content that is either an episode of a series (e.g., pilot for series) or is non-series content such as a movie (e.g., Titanic). Note that a program may also link to a title (e.g., "Weekend Movie Block") or may not have any title linkage (e.g., "6 PM News").

***/Content/@version:** The version attribute shall be "Title".

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@idType:** The idType attribute shall be "TitleId". This represents the technical key for the title.

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@authorativeSource:** The authorative source shall be "GlobalId". The global identifier is a common source that is shared by all applications.

10.1.1.1.3 Version

A version is the logical slicing of title information. For example, program "Sex in the City" may have multiple versions for rating purposes (e.g., TV-14 vs. TV-MA), segments, language, wide screen/full screen, length, etc.

***/Content/@version:** The version attribute shall be "Version".

***/Content/ProgramContent/ContentMetaData/ContentId/HouseNumber:** The house number element shall be specified. This represents the business key for the version.

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@idType:** The idType attribute shall be “versionId”. This represents the technical key for the version.

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@authorativeSource:** The authorative source shall be “GlobalId”. The global identifier is a common source that is shared by all applications

10.1.1.1.4 Element (Segment)

An element is the logical descriptor for timing and media information within a version (e.g., 1, 2, 3). All elements are defined within a version element in BXF.

***/Content/ProgramContent/Elements/Element/ProgramElement/SegmentNumber:** Shall identify the segment within the version. This represents the business key for the segment.

***/Content/ProgramContent/Elements/Element/ProgramElement ContentMetaData/ContentId/AlternateId/@idType:** The idType attribute shall be “mediaId”. This represents the technical key for the segment.

***/Content/ProgramContent/Elements/Element/ProgramElement ContentMetaData/ContentId/AlternateId/@authorativeSource:** The authorative source shall be “GlobalId”. The global identifier is a common source that is shared by all applications

10.1.1.1.5 Media

Media is a container for clips (e.g., TAPE1, Server3). Media is identified within a segment in BXF. The location of the media identifier depends on the type of media.

***/Content/ProgramContent/Elements/Element/ProgramElement/ContentMetaData/Media/MediaLocation/Location/PhysicalAssetName/Media/ReferenceName:** The media reference name shall be used to identify the media when the clip is located on a physical asset. This represents the business key for the media.

***/Content/ProgramContent/Elements/Element/ProgramElement/ContentMetaData/Media/MediaLocation/Location/AssetServer/ReferenceName:** The asset server reference name shall be used to identify the server when the clip is located on a server. This represents the business key for the media.

***/Content/ProgramContent/Elements/Element/ProgramElement/ContentMetaData/Media/MediaLocation/Location/RouterSource/Name:** The router source name shall be used to identify the source when the clip is accessed through a router source. This represents the business key for the media.

10.1.1.1.6 Clip

A clip is what airs/plays. Clips are identified within a segment in BXF.

***/Content/ProgramContent/Elements/Element/ProgramElement ContentMetaData/ContentId/HouseNumber:** The house number element shall be specified. This represents the business key for the clip.

10.1.1.1.7 Program Only Transactions

Since program schedules are typically built in stages, there is the need to send ‘program only’ messages that tell other systems about program content without necessarily describing the attributes of the program In this case, it describes only the programs that will air without the details of the titles and media versions that will

drive the automation during the actual airing of the content. Example uses for program only transactions include sending a schedule planning grid or information for electronic program guides.

10.1.2 Program Content Associations (Normative)

The previous sections described how content is uniquely identified at each level. A similar approach is used to associate content at one level to content at another level. The following associations are recommended:

- A series or program can have titles, or may exist without titles (e.g., “Program Only Grid”)
- A title can have versions
- A version can have segments, media and clips

10.1.2.1 Associating a Series to a Title

For series related content, the series shall be identified in the Content element that contains the associated title, as indicated by the following title version attribute and series identifier. The series identifier shall be provided in addition to any title identifiers that are specified. Refer to the message examples for additional information.

***/Content/@version:** The version attribute shall be “Title”.

[Title identifiers included here]

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@idType:** The idType attribute shall be “SeriesId”. This represents the technical key for the series.

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@authorativeSource:** The authorative source shall be “GlobalId”. The global identifier is a common source that is shared by all applications.

10.1.2.2 Associating a Title to a Version

The title shall be identified in the Content element that contains the associated version, as indicated by the following title version attribute and title identifier. The title identifier shall be provided in addition to any version identifiers that are specified. Refer to the message examples for additional information.

***/Content/@version:** The version attribute shall be “Version”.

[Version identifiers included here]

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@idType:** The idType attribute shall be “TitleId”. This represents the technical key for the title.

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@authorativeSource:** The authorative source shall be “GlobalId”. The global identifier is a common source that is shared by all applications.

10.1.2.3 Associating a Version to Segment, Media and Clip Information

Segment, media and clip information are defined in the Elements element in BXF, which is a sub-element of the Content element. This makes the association to a version contextual, since segment, media and clip information are defined within a version.

10.1.3 Non-Program Content (Normative)

Non-program content is content that is typically short in nature and is scheduled in breaks between the segments of a program, such as commercials, promos, ids, etc.

***/Content/@version:** The version attribute shall be “NonProgram”.

***/Content/ProgramContent/ContentMetaData/ContentId/HouseNumber:** The house number element shall be specified. This represents the business key for the non-program content.

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@idType:** The idType attribute shall be “NonProgramId”. This represents the technical key for the non-program content.

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@authorativeSource:** The authorative source shall be “GlobalId”. The global identifier is a common source that is shared by all applications.

10.1.4 Unknown Content (Normative)

Unknown content is typically media related, where an application does not know if the content information is program or non-program content. Note that the */Content/ProgramContent element is utilized even though the content may be non-program content.

/Content/@version:** The version attribute shall be set to “BxfUnknown***”.

***/Content/ProgramContent/ContentMetaData/ContentId/HouseNumber:** The house number element shall be specified. This represents the business key for the program or non-program content.

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@idType:** The idType attribute shall be “MediaId”. This represents the technical key for the program or non-program content.

***/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@authorativeSource:** The authorative source shall be “GlobalId”. The global identifier is a common source that is shared by all applications

10.1.5 Content Processing Conventions (Normative)

The following conventions apply to all content messages.

Recommended Conventions – Content

BxfMessage/BxfData/Content/@contentExists: This attribute should not be used in a content message.

BxfMessage/BxfData/Content/@timestamp: All date, time and timestamp values in a content message shall use the canonical representation of UTC (e.g., 2007-02-01T15:30:00.0Z) with no time zone offset. This attribute identifies when content was created, updated or deleted. The content creation time may differ from when the content message is actually sent to other applications. This value should be included if it is available.

BxfMessage/BxfData/Content/@user: Identifies the user that created, updated or deleted the content. This user may differ from the user that causes a content message to be sent to other applications. This value should be included if it is available.

BxfMessage/BxfData/Content/@version: The version attribute shall be provided within a content message to identify the content level. The valid values for version are: “Series”, “Title”, “Version”, “NonProgram” and “***BxfUnknown***”.

10.1.5.1 Content Metadata

***/ContentMetaData/ContentId:** One or more content identifiers shall be specified, as well as any content associations (e.g., Title that is associated to Version).

***/ContentMetaData/Name:** One or more names should be specified.

***/ContentMetaData/Genre:** One or more genres should be specified.

***/ContentMetaData/Description:** One or more descriptions should be specified.

10.1.5.2 Content Metadata – Media

BxfMessage/BxfData/Content/ProgramContent/ContentMetaData/Media: Media information that is applicable to all media shall be provided at this level. The element level shall be used to describe information specific to the element, and shall override information provided at upper levels. Media information shall be provided at the element level (i.e., BxfMessage/BxfData/Content/ProgramContent/Elements/Element/ ProgramElement/ContentMetaData/Media).

***/ContentMetaData/Media/Location/MediaLocation/AssetServer/PathName:** The path name shall include the filename. If the path is not known, then an empty element shall be specified.

***/ContentMetaData/Media/MediaLocation/Location/PhysicalAsset:** If the media reference identifier is known, but the type of location is not known, then the MediaReferenceName within element should be used to specify the reference identifier.

10.1.5.3 Program Content – Elements

***/ProgramContent/Elements/Element:**

***/ProgramContent/Elements/Element/Offset:** This is the offset from start of program (i.e., not relevant to the material location). Systems shall set "Offset" to 00:00:00:00 when value is not applicable.

***/ProgramContent/Elements/Element/Duration:** Duration should be ignored or used for informational purposes (i.e., not relevant to the material location).

***/ProgramContent/Elements/Element/ProgramElement/ContentMetadata/ContentId/HouseNumber:** The "HouseNumber" element at the element level shall contain the media/clip ID.

***/ProgramContent/Elements/Element/ProgramElement/ContentMetaData/ContentId/AlternateId/@idType:** The idType attribute shall be “mediaId”. This represents the technical key for the segment.

***/ProgramContent/Elements/Element/ProgramElement/ContentMetaData/ContentId/AlternateId/@authorativeSource:** The authorative source shall be “GlobalId”. The global identifier is a common source that is shared by all applications

***/ProgramContent/Elements/Element/ProgramElement/ContentMetadata/Media/Choice:** When a system does not know if the media is Baseband, Pre-compressed, or Profile macro, then an empty Baseband element shall be added (e.g., <BaseBand/>).

/ProgramContent/Elements/Element/ProgramElement/ContentMetadata/Media/MediaLocation/PhysicalAsset/@assetName:** The “assetName” shall identify the type of asset. For example: Tape, CD, Server. A value of “BxfUnknown***” shall be used when the assetName is not known.

***/ProgramContent/Elements/Element/ProgramElement/ContentMetadata/Media/MediaLocation/Location/Satellite/Transponder:** Satellite Transponder is a mandatory field. If it is unknown it shall be set to zero. Elements include:

- ***/ProgramContent/Elements/Element/ProgramElement/ContentMetadata/Media/MediaLocation/Location/Satellite/Transponder/Polarity:** The “Polarity” element should be ignored if Transponder is set to zero.

- ***/ProgramContent/Elements/Element/ProgramElement/ContentMetadata/Media/MediaLocation/Location/Satellite/Transponder/TransponderNumber:** The “TransponderNumber” element should be is set to zero if unknown.
- ***/ProgramContent/Elements/Element/ProgramElement/ContentMetadata/Media/MediaLocation/Location/Satellite/Transponder/Receiver:** The “Receiver” element should be is set to ***BxfUnknown*** if unknown.
- ***/ProgramContent/Elements/Element/ProgramElement/ContentMetadata/Media/MediaLocation/Location/Satellite/Transponder/Encoder:** The “Encoder” element should be set to ***BxfUnknown*** if unknown.

10.1.5.4 Program Content – Series

***/ProgramContent/Series:** The "Series" element may be included for episodic programs (i.e., mini-series, series, etc.) to identify the season for an episode. Most of the elements in Series are redundant, since this information is provided in */ProgramContent/ContentMetaData. Attributes and elements include:

- ***/ProgramContent/Series/@seriesId:** The “seriesId" attribute shall uniquely identify the series.
- ***/ProgramContent/Series/EpisodeName:** The “episode name” attribute shall contain the title/episode name.

10.1.5.5 Program Content – Parental Rating

***/ProgramContent/ParentalRating:** Parental ratings may be provided at one or more levels (e.g., Version, Title, etc.). Attributes and elements of ParentalRating are:

- ***/ProgramContent/ParentalRating/@region:** The "region" attribute shall contain the mapped region identifier (Configuration type: ParentalRatingRegion).
- ***/ProgramContent/ParentalRating/Rating/@dimension:** The "dimension" attribute shall contain the mapped dimension value (Configuration type: ParentalRatingDimension). Examples: TV Parental.
- ***/ProgramContent/ParentalRating/Rating/@value:** The "value" attribute shall contain the mapped value for the dimension (Configuration type: ParentalRatingValue). Examples: TV-PG, TV-G, R.
- ***/ProgramContent/ParentalRating/ContentAdvisory:** The "ContentAdvisory" element shall contain the mapped content advisory value (Configuration type: ParentalRatingContentAdvisory). It includes an attribute of language. Examples: Dialogue, Some Sexual Situations, Harsh Language.

10.2 Requesting Content Information (Queries) (Normative)

The following sections identify how an application can request the content of another application. Each section is annotated, indicating if the functionality is required, optional or to be avoided.

10.2.1 Obtaining All Content Information

The conventions for handling BXF queries for content information are described in the following sections. Some query capabilities are recommended.

10.2.1.1 Request Message

Applications shall provide the capability to consume a request for all of their content values.

Applications may provide the capability to create a request message to get all of another application's content values, as well as the functionality to consume the corresponding reply.

The example message contains a request for all content values:

Content – Get All

Recommended Conventions – Content Query Request

BxfMessage/@ext:usage: The usage shall be "Content Query".

BxfMessage/@messageType: The message type shall be "Request".

BxfQuery/Where: For Content Query Request, the <WhereClause> element for "Content ="*"' shall be supported, which indicates that all content values are being requested.

BxfMessage/BxfQuery: Content query request capabilities should be implemented. If this capability is not provided by an application, then the "not_supported" error type should be utilized on acknowledgement messages.

10.2.1.2 Reply Message

Applications shall provide the capability to reply to content query requests. This can be achieved through a reply that contains the content data or through a reply that references one or more file locations where the content data resides. For example:

Content – Get All Reply (data within reply message)
Content – Get All Reply (reply with file reference)
Content – Get All Reply (no data exists)

Recommended Conventions – Content Query Reply

BxfMessage/@ext:usage: The usage shall be "Content Query Reply".

BxfMessage/@messageType: The message type shall be "Reply".

BxfMessage/BxfQueryResponse: Content query response capabilities should be implemented. If this capability is not provided by an application, then the "not_supported" error type should be utilized on acknowledgement messages.

BxfMessage/BxfQueryResponse: The reply message for a "get content" request should utilize the file-based approach (described below) if the message size exceeds 2 MB.

Additional conventions are specified in **Reply Messages**.

10.2.2 Obtaining Type-Specific Content Values

The conventions for handling BXF queries for type-specific content information are described in the following sections. At least one of the recommended request messages is required.

10.2.2.1 Request Message

Applications shall provide the capability to consume a request for type-specific content values. Applications should support a query by GUID. Other query types are specified below.

Additional type-specific queries are possible (e.g., UMID, client-specific, etc.) and may be implemented based on an application's capabilities. Note that the use of alternate IDs requires a mapping between identifier types within the configuration data.

The following examples show the different types of query requests that are recommended:

Content – Get Program by Alternate ID (GUID)
Content – Get Program by Name
Content – Get Series by Alternate ID (GUID)
Content – Get Series by Name
Content – Get Title by Alternate ID (GUID)
Content – Get Title by Name and Series ID (GUID)
Content – Get Title by ISAN
Content – Get Version by Alternate ID (GUID)
Content – Get Version by House Number
Content – Get Media by House Number
Content – Get Non-program by Alternate ID (GUID)
Content – Get Non-program by Name
Content – Get Non-program by ISCI

Recommended Conventions – Type-Specific Content Query Request

BxfMessage/@ext:usage: The usage shall be "Content Query".

BxfMessage/@messageType: The message type shall be "Request".

For Type-Specific Content Query Request, the where clause should contain one of the following:

- **Content /@version:** A content level, such as Series, Title or Version.
- **Content/*/ContentMetadata/ContentId:** A content ID such as a house number, ISAN or alternate identifier.
- **Content/*/ContentMetadata/Name:** A series, title or version name.

Applications shall support the "=" operator and the "and" operator; all other operators are optional.

10.2.2.2 Reply Message

Applications shall provide the capability to reply to a type-specific configuration query request. For example:

Content – Get Program by Alternate ID (GUID) Reply
Content – Get Series by Alternate ID (GUID) Reply
Content – Get Title by Alternate ID (GUID) Reply
Content – Get Title by Name and Series ID (GUID) Reply
Content – Get Version by Alternate ID (GUID) Reply
Content – Get Version by House Number Reply
Content – Get Media by House Number Reply
Content – Get Non-program by Alternate ID (GUID) Reply
Content – Get Non-program by Name Reply

Recommended Conventions – Type-Specific Content Query Reply

A reply message should contain content for the requested level and all content beneath it. For example, a reply to a request for a series should contain the series, all titles within that series, all versions within the titles, etc.; whereas, a reply to a request for a version should contain the version, its segments and its media.

BxfMessage/@ext:usage: The usage shall be “Content Query Reply”.

BxfMessage/@messageType: The message type shall be “Reply”.

Additional conventions are specified in **Reply Messages**.

10.2.3 Obtaining Content Values through Advanced Queries

The BXF protocol supports advanced querying capabilities and includes conditional logic (e.g., “and/or” logic) and operators (e.g., <, >=, etc.). Applications may provide support for advanced operations other than the get type-specific examples that were described in the previous section, but these are not recommended.

In the context of content, these capabilities should be avoided, since they will require significant time and effort to develop.

10.3 Notifying Others when Content is Added (Normative)

10.3.1 Program Content

The following examples show the different types of program content additions:

Content – Add Program
Content – Add Series
Content – Add Series (with Title)
Content – Add Series (with Title and Versions)
Content – Add Movie Title (with Versions)
Content – Add Multiple Movie Titles

Recommended Conventions – Add Program Content

The highest content levels (e.g., program/series and title (for movies)) shall be established first, since lower content levels will reference these levels. For series-oriented content, title and version information may be provided within a add series message, or this information may be provided in separate messages that follow the add series message. Similarly, for movie titles, the title information may include version information, or the version information may be sent in separate messages to follow the add title message.

BxfMessage/@ext:usage: The usage shall be “Content”.

BxfMessage/@action: The action shall be “add”.

BxfMessage/@messageType: The message type shall be “Information”.

BxfMessage/BxfData/Content/@version: The version shall be “Series”, “Title” or “Version”.

10.3.2 Non-Program Content

The following examples show the different types of non-program content additions:

Content – Add Non-Program Content
Content – Add Non-Program with Unknown Spot Type
Content – Add Non-Program with One Spot on Tape with Multiple Spots

Recommended Conventions – Add Non-Program Content

BxfMessage/@ext:usage: The usage shall be “Content”.

BxfMessage/@action: The action shall be “add”

BxfMessage/@messageType: The message type shall be “Information”.

BxfMessage/BxfData/Content/@version: The version shall be “NonProgram”.

BxfMessage/BxfData/Content/NonProgramContent/Details/SpotType: For Non-Program Content, the spot type shall contain a value from the SpotType configuration information.

10.3.3 Media for Unknown Content

The following example shows a media addition where the content type is not known:

Content – Add Media for Unknown Content

Recommended Conventions – Add Media for Unknown Content

BxfMessage/@ext:usage: The usage shall be “Media”.

BxfMessage/@action: The action shall be “add”

BxfMessage/@messageType: The message type shall be “Information”.

BxfMessage/BxfData/Content/@version: The version shall be “***BxfUnknown***”.

BxfMessage/BxfData/Content/ProgramContent/*: Media information shall be provided under the ProgramContent structure even though content may actually be non-program content.

10.4 Notifying Others when Content is Updated (Normative)

Content update messages may apply to one or more of the content levels. The messages should be limited to the unique identifiers and the information that changed. For example, if a description is added to a version, then only the version-related content should be sent with the appropriate identifiers – The message creator does not have to include information about the associated title and series (if applicable), although this information may be provided.

10.4.1 Program Content

The following examples show the different types of program content updates:

Content – Update Program (Add Genre)
Content – Update Series (Add Genre)
Content – Update Title (Add Parental Ratings)
Content – Update Title (Add Media)
Content – Update Version (Add/Update Descriptions)
Content – Update Version (Add Segment)

Recommended Conventions – Update Program Content

BxfMessage/@ext:usage: The usage shall be “Content”.

BxfMessage/@action: The action shall be “update”.

BxfMessage/@messageType: The message type shall be “Information”.

10.4.2 Non-Program Content

The following examples show the different types of non-program content updates:

Content – Update Non-Program (Add/Update Description)
Content – Update Non-Program (Add Media)

Recommended Conventions – Update Non-Program Content

BxfMessage/@ext:usage: The usage shall be “Content”.

BxfMessage/@action: The action shall be “update”.

BxfMessage/@messageType: The message type shall be “Information”.

10.5 Notifying Others when Content is Removed (Normative)

The context for this message is to let an application know that another application has removed content metadata. (For removal of the actual essence of the content, the “Content Transfer” message type is used with the extension usage of “Purge List” and the transfer type of “Purge”). This is not a request to remove the data; it is a notification that a system removed the data. It is up to the receiver of this message to determine if the content can be removed following their own business rules. Depending on the action taken in the application, the acknowledgement should reflect that action. If the content was not removed from the application receiving the message, acknowledgement should be set to “Warning” with a reason of why it was not removed.

10.5.1 Program Content

The following examples show the different types of program content removals:

Content – Remove Program
Content – Remove Series
Content – Remove Title
Content – Remove Version

Recommended Conventions – Remove Program Content

BxfMessage/@ext:usage: The usage shall be “Content”.

BxfMessage/@action: The action shall be “remove”.

BxfMessage/@messageType: The message type shall be “Information”.

BxfMessage/BxfData/Content/@version: The version shall be either “Series”, “Title” or “Version”.

10.5.2 Non-Program Content

The following example shows a non-program content removal:

Content – Remove Non-Program

Recommended Conventions – Remove Non-Program Content

BxfMessage/@ext:usage: The usage shall be “Content”.

BxfMessage/@action: The action shall be “remove”.

BxfMessage/@messageType: The message type shall be “Information”.

BxfMessage/BxfData/Content/@version: The version shall be “NonProgram”.

11 Content Transfer

11.1 Overview (Informative)

The ContentTransfer element within the BXF protocol allows applications to request that content-related activities be completed in other (primarily automation) applications. BXF definition:

> *"The movement or copying of content from one location to another using any of several possible methodologies including the physical movement of tapes that have the content recorded on them, real-time recording schedules, or movement of the content over an IP network."*

11.1.1 Content Transfer Processing Conventions (Normative)

It is important to note that content transfer messages are used to manipulate essence (e.g., copy, purge, etc.). The content metadata changes that result from content transfer activities should be provided in Content messages, not Content Transfer messages. For example, when content is dubbed based on a content transfer message, a Content Transfer Status message should get generated that indicates the status of the transfer request (e.g., Completed, Failed, etc.). In addition, a Content message should get generated that contains metadata information, such as the timing information for the content that was processed.

The following conventions apply to all content transfer messages.

Recommended Conventions – Content Transfer

BxfMessage/BxfData/@action: No action should be specified on content transfer messages.

BxfMessage/BxfData/ContentTransfer/@priority: Priority should be provided (Normal is the default).

BxfMessage/BxfData/ContentTransfer/Source: The source should be provided if known.

BxfMessage/BxfData/ContentTransfer/Destination: The destination should be provided if known.

11.1.1.1 Program Content

For Content Transfer, minimal content information should be sent, enough to visually recognize the content (e.g., media/clip Id and version, title and/or series name).

Recommended Conventions – Content Transfer for Program Content

BxfMessage/BxfData/ContentTransfer/Content/@level: The content level should be "Media".

BxfMessage/BxfData/ContentTransfer/Content/ProgramContent/ContentMetadata/ContentId/HouseNumber: For Content Transfer, the Content ID shall contain the "HouseNumber" element. House number refers to the media/clip ID.

BxfMessage/BxfData/ContentTransfer/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@idType: An alternate identifier should be provided to identify the media, where the idType attribute shall be "MediaId". This represents the technical key for the program content.

BxfMessage/BxfData/ContentTransfer/Content/ProgramContent/ContentMetaData/ContentId/AlternateId/@idType: Additional alternate identifiers may be provided to identify the version, title and series, where the idType attribute shall be "VersionId", "TitleId" and "SeriesId" respectively. These identifiers provide associations to the media.

BxfMessage/BxfData/ContentTransfer/Content/ProgramContent/ContentMetadata/Name: For Content Transfer, the Content Name element may be included. It contains the version name. The type attribute may be used to provide the version, title and series names.

- */Name/@type="VersionName": The name element shall contain the version name.
- */Name/@type="TitleName": The name element shall contain the title name.
- */Name/@type="SeriesName": The name element shall contain the series name.

11.1.1.2 Non-Program Content

For Content Transfer, minimal content information should be sent, enough to visually recognize the content (e.g., Media Id and name).

Recommended Conventions – Content Transfer for Non-Program Content

BxfMessage/BxfData/ContentTransfer/Content/@level: The content level should be "NonProgram".

BxfMessage/BxfData/ContentTransfer/Content/NonProgramContent/ContentMetadata/ContentId/HouseNumber: For Content Transfer, the Content ID shall contain the "HouseNumber" element. House Number refers to the material/clip ID.

BxfMessage/BxfData/ContentTransfer/Content/NonProgramContent/ContentMetaData/ContentId/AlternateId/@idType: An alternate identifier should be provided to identify the media, where the idType attribute is "MediaId". This represents the technical key for the non-program content.

BxfMessage/BxfData/ContentTransfer/Content/NonProgramContent/ContentMetadata/Name: For Content Transfer, the Content Name element may be included. It contains the non-program content name.

11.2 Content Transfer Status Query (Normative)

Content transfer status query is used to query an application regarding the status of a content transfer. Applications may provide this capability, but it is not recommended.

11.3 Dub Order (Normative)

BXF definition:

> *"A set of instructions that cause the specified content, which is generally a physical asset, to be copied from its original source material and location to a new destination."*

The following example shows a dub list (order):

Content Transfer – Dub List (Program)
Content Transfer – Dub List (Non-Program)

Recommended Conventions – Dub Order

BxfMessage/@ext:usage: The usage shall be "Dub List".

BxfMessage/@messageType: The message type shall be "Information".

BxfMessage/BxfData/ContentTransfer/@transferType: For Dub Order, the "transferType" shall be "Duplication".

BxfMessage/BxfData/ContentTransfer/Source: For Dub Order, the "Source" element may be included to identify the source.

BxfMessage/BxfData/ContentTransfer/Source/Choice: For Dub Order, the choice between Baseband and Pre-compressed may be sent. If unknown, then an empty Baseband element should be sent.

BxfMessage/BxfData/ContentTransfer/Destination: For Dub Order, the "Destination" element may be included to identify the destination.

BxfMessage/BxfData/ContentTransfer/Destination/Choice: For Dub Order, the choice between Baseband and Pre-compressed may be sent. If unknown, then an empty Baseband element should be sent.

BxfMessage/BxfData/ContentTransfer/Destination/Media/MediaLocation/Location/AssetServer/PathName: For Dub Order, the "PathName" element may be empty to indicate that the destination is either unknown or at the discretion of the message consumer.

BxfMessage/BxfData/ContentTransfer/Destination/Media/MediaLocation/Location/ArchiveGroup: For Dub Order, the "ArchiveGroup" element may be included.

BxfMessage/BxfData/ContentTransfer/Destination/UsagePolicy: For Dub Order, the "UsagePolicy" element may be included.

11.4 Purge Order (Normative)

BXF definition:

> *"A set of instructions that determines when a system should delete content from its current location."*

If an originator needs to revise a previously transmitted purge order that has not been purged (e.g., get new copy instructions for content that is targeted for purge in the near future) then it should send another purge order for the same content with updated purge instructions. The destination should process the revised purge order and use it instead.

The following examples show purge orders:

Content Transfer – Purge List (Program)
Content Transfer – Purge List (Non-Program)
Content Transfer – Purge List (One Media Instance)

Recommended Conventions – Purge Order

BxfMessage/@ext:usage: The usage shall be "Purge List".

BxfMessage/@messageType: The message type shall be "Information".

BxfMessage/BxfData/ContentTransfer/@transferType: For Purge Order, the "transferType" shall be "Purge".

11.5 Record Order (Normative)

BXF definition:

> *"A set of instructions that determines when a system expects to record content being sent to it typically via satellite. The instructions include when the recording is scheduled to take place and what source to use as well as optionally where to record it."*

The following examples show record orders:

Content Transfer – Record off Satellite (Program)
Content Transfer – Record off Satellite (Non-Program)

Recommended Conventions – Record Order

BxfMessage/@ext:usage: The usage shall be “Record List”.

BxfMessage/@messageType: The message type shall be “Information”.

BxfMessage/BxfData/ContentTransfer/@transferType: For Record Order, the “transferType” shall be “Recording”.

BxfMessage/BxfData/ContentTransfer/Source: For Record Order, the “Source” element may be included to identify the source.

BxfMessage/BxfData/ContentTransfer/Source/Media/MediaLocation/Location/Satellite/Transponder/TransponderNumber: It the transponder number is not known then a value of zero should be specified.

BxfMessage/BxfData/ContentTransfer/Source/Media/MediaLocation/Location/Satellite/Receiver: It the receiver is not known then “***BxfUnknown***” should be specified.

BxfMessage/BxfData/ContentTransfer/Source/Media/MediaLocation/Location/Satellite/Encoder: It the encoder is not known then “***BxfUnknown***” should be specified.

BxfMessage/BxfData/ContentTransfer/Destination/Media/MediaLocation: For Record Order, the “MediaLocation” element may be included.

11.6 Transfer Order (Normative)

A set of instructions requesting that content be transferred from one location to another.

The following examples show transfer orders:

Content Transfer – Transfer Order (Program)
Content Transfer – Transfer Order (Non-Program)

Recommended Conventions – Transfer Order

BxfMessage/@ext:usage: The usage shall be “File Transfer”.

BxfMessage/@messageType: The message type shall be “Information”.

BxfMessage/BxfData/ContentTransfer/@transferType: For Transfer Order, the “transferType” shall be “File transfer”.

BxfMessage/BxfData/ContentTransfer/Source: ForTransfer Order, the “Source” element should be included to identify the source.

BxfMessage/BxfData/ContentTransfer/Destination: For Transfer Order, the “Destination” element should be included to identify the destination.

11.7 Content Transfer Status (Normative)

Applications that consume Content Transfer messages should generate Content Transfer Status messages to indicate the status of the transfer.

The following examples show content transfer status:

Content Transfer Status (Completed)
Content Transfer Status (Failed)

Recommended Conventions – Content Transfer Status

BxfMessage/@ext:usage: The usage shall be “Content Transfer Status”.

BxfMessage/@messageType: The message type shall be “Information”.

BxfMessage/BxfData/ContentTransfer/@status: Shall be the status of the content transfer.

BxfMessage/BxfData/ContentTransfer/@statusDescription: Shall be the description of the content transfer status.

12 Format

12.1 Overview (Informative)

The Format element within the BXF protocol allows applications to share program format information. Definition within BXF:

> *“The definition of the empty structure of a program as used by the traffic system to construct a schedule grid”*

The format-related messages support the following capabilities:

- The capability to request format-related information.
- The capability to reply to requests for format-related information including “all” formats and/or targeted content information that exists within an application.
- The capability to generate and consume notification messages relating to format additions, updates and removals. These messages can originate from any participating application.

When adding a format, all elements should be included; however, on update or removal of elements, only those that have changes should be included. The diagrams below illustrate this concept.

Step 1: Add Format:

Seq.	Type	TimeCode	Duration	Description	ID	InsertAfterID	Action Notes
1	Segment	00:00:00;00	00:33:15;00	Segment 1	3E072176-B4E6-402e-BCBE-17A6F1F7232A		
2	Comment	00:33:15;00	00:00:00;00	Break 1	3E072176-B4E6-402e-BCBE-17A6F1F7245A	3E072176-B4E6-402e-BCBE-17A6F1F7232A	
3	Break	00:33:15;00	00:03:00;00	Break	2E072176-B4E6-672e-BCBE-17A6F1F7245A	3E072176-B4E6-402e-BCBE-17A6F1F7245A	
4	Non-commercial Break	00:36:15;00	00:00:30;00	Promo	2E072176-B4E6-672e-BCBE-17A6F1F7345C	2E072176-B4E6-672e-BCBE-17A6F1F7245A	
5	Segment	00:36:45;00	00:20:10;00	Segment 2	3E072176-B4E6-402e-BCBE-17A6F1F7235F	2E072176-B4E6-672e-BCBE-17A6F1F7345C	
6	Comment	00:56:55;00	00:00:00;00	Break 2	3E072176-B4E6-405e-BCBE-17A6F1F7245B	3E072176-B4E6-402e-BCBE-17A6F1F7235F	
7	Break	00:56:55;00	00:02:30;00	Break	2E072176-B4E6-672e-BCBE-17A6F1F7246D	3E072176-B4E6-405e-BCBE-17A6F1F7245B	
8	Non-commercial Break	00:59:25;00	00:00:30;00	Promo	2E072176-B4E6-672e-BCBE-17A6F1F7334D	2E072176-B4E6-672e-BCBE-17A6F1F7246D	
9	Non-commercial Break	00:59:55;00	00:00:05;00	ID	2E072176-B4E6-672e-ACBE-17A6F1F7335E	2E072176-B4E6-672e-BCBE-17A6F1F7334D	

Step 2: Update Format (Change Timecode and Durations)

In this step, the duration is updated on the third element (Sequence 3, Break), and the timecode and duration are updated on the fourth element (Sequence 4, Non-commercial Break).

Seq.	Type	TimeCode	Duration	Description	ID	InsertAfterID	Action Notes
1	Segment	00:00:00;00	00:33:15;00	Segment 1	3E072176-B4E6-402e-BCBE-17A6F1F7232A		
2	Comment	00:33:15;00	00:00:00;00	Break 1	3E072176-B4E6-402e-BCBE-17A6F1F7245A	3E072176-B4E6-402e-BCBE-17A6F1F7232A	
3	Break	00:33:15;00	00:02:30;00	Break	2E072176-B4E6-672e-BCBE-17A6F1F7245A	3E072176-B4E6-402e-BCBE-17A6F1F7245A	Changed Duration
4	Non-commercial Break	00:35:45;00	00:01:00;00	Promo	2E072176-B4E6-672e-BCBE-17A6F1F7345C	2E072176-B4E6-672e-BCBE-17A6F1F7245A	Changed Timecode and Duration
5	Segment	00:36:45;00	00:20:10;00	Segment 2	3E072176-B4E6-402e-BCBE-17A6F1F7235F	2E072176-B4E6-672e-BCBE-17A6F1F7345C	
6	Comment	00:56:55;00	00:00:00;00	Break 2	3E072176-B4E6-405e-BCBE-17A6F1F7245B	3E072176-B4E6-402e-BCBE-17A6F1F7235F	
7	Break	00:56:55;00	00:02:30;00	Break	2E072176-B4E6-672e-BCBE-17A6F1F7246D	3E072176-B4E6-405e-BCBE-17A6F1F7245B	
8	Non-commercial Break	00:59:25;00	00:00:30;00	Promo	2E072176-B4E6-672e-BCBE-17A6F1F7334D	2E072176-B4E6-672e-BCBE-17A6F1F7246D	
9	Non-commercial Break	00:59:55;00	00:00:05;00	ID	2E072176-B4E6-672e-ACBE-17A6F1F7335E	2E072176-B4E6-672e-BCBE-17A6F1F7334D	

Step 3: Delete Format Element and Add New Format Element

In this step, third element (Sequence 3, Break) is removed, the fourth element becomes the third element (Sequence 3, Non-commercial Break) and a new fourth element is added (Sequence 4, Break). The fifth element (Sequence 5, Segment) is updated based on the new fourth element.

Seq.	Type	TimeCode	Duration	Description	ID	InsertAfterID	Action Notes
1	Segment	00:00:00;00	00:33:15;00	Segment 1	3E072176-B4E6-402e-BCBE-17A6F1F7232A		
2	Comment	00:33:15;00	00:00:00;00	Break 1	3E072176-B4E6-402e-BCBE-17A6F1F7245A	3E072176-B4E6-402e-BCBE-17A6F1F7232A	
~~3~~	~~Break~~	~~00:33:15;00~~	~~00:02:30;00~~	~~Break~~	~~2E072176-B4E6-672e-BCBE-17A6F1F7245A~~	~~3E072176-B4E6-402e-BCBE-17A6F1F7245A~~	Deleted line
3	Non-commercial Break	00:33:15;00	00:01:00;00	Promo	2E072176-B4E6-672e-BCBE-17A6F1F7345C	3E072176-B4E6-402e-BCBE-17A6F1F7245A	Changed line
4	Break	00:34:15;00	00:02:30;00	Break	5E072176-B4E6-672e-ACBE-17A6F1F7345F	2E072176-B4E6-672e-BCBE-17A6F1F7345C	Added line
5	Segment	00:36:45;00	00:20:10;00	Segment 2	3E072176-B4E6-402e-BCBE-17A6F1F7235F	5E072176-B4E6-672e-ACBE-17A6F1F7345F	Change due to added line above
6	Comment	00:56:55;00	00:00:00;00	Break 2	3E072176-B4E6-405e-BCBE-17A6F1F7245B	3E072176-B4E6-402e-BCBE-17A6F1F7235F	
7	Break	00:56:55;00	00:02:30;00	Break	2E072176-B4E6-672e-BCBE-17A6F1F7246D	3E072176-B4E6-405e-BCBE-17A6F1F7245B	
8	Non-commercial Break	00:59:25;00	00:00:30;00	Promo	2E072176-B4E6-672e-BCBE-17A6F1F7334D	2E072176-B4E6-672e-BCBE-17A6F1F7246D	
9	Non-commercial Break	00:59:55;00	00:00:05;00	ID	2E072176-B4E6-672e-ACBE-17A6F1F7335E	2E072176-B4E6-672e-BCBE-17A6F1F7334D	

This approach makes format processing similar to schedule processing, where events are linked by event identifiers. In the case of formats, elements are linked using the “PrimaryElementId” element and the “InsertAfterElementId” extension element. This provides the capability to make precise changes to a format as opposed to resending the format when elements are added or removed.

12.1.1 Format Conventions (Normative)

When adding a format, all elements should be included; however, on update or removal of elements, only those that have changes should be included.

Recommended Conventions – Format

BxfMessage/BxfData/Format/FormatId: The “FormatId” element is represented using a GUID.

BxfMessage/BxfData/Format/FormatLength: The “FormatLength” element is represented using the SmpteTimeCode format.

BxfMessage/BxfData/Format/FormatUsage: The optional “FormatUsage” element identifies the channels and dates the format can be used.

BxfMessage/BxfData/Format/FormatStructure/FormatElements/PrimaryElementId: The PrimaryElementId element is represented using a GUID.

BxfMessage/BxfData/Format/FormatStructure/FormatElements/ElementType: The ElementType element is an enumeration: Segment, Comment, Break, Non-commercial Break or ID.

BxfMessage/BxfData/Format/FormatStructure/FormatElements/FormatOrderSequence: The FormatOrderSequence element may be provided to indicate the element order within the sequence. It should only be used placement when the InsertAfterElementId (see below) does not exist of is not recognized.

BxfMessage/BxfData/Format/FormatStructure/FormatElements/PrimaryDuration: The PrimaryDuration element shall be represented using the SmpteTimeCode format.

BxfMessage/BxfData/Format/FormatStructure/FormatElements/NonPrimaryElements: Secondary events to the primary event are defined within the primary structure under the NonPrimaryElements element.

BxfMessage/BxfData/Format/FormatStructure/FormatElements/NonPrimaryElements/NonPrimaryDuration: The NonPrimaryDuration element shall be represented using the SmpteTimeCode format.

BxfMessage/BxfData/Format/FormatStructure/FormatElements/PrivateInformation/BreakCode: These user-defined codes allow systems to configure breaks for certain types of spots. BreakCode is an extension configuration type (CONF) that allows systems to indicate the break codes that can be associated with breaks in a format.

BxfMessage/BxfData/Format/FormatStructure/FormatElements/PrivateInformation/ExcludeProductCodes: ExcludeProductCodes extension element is a configuration type (CONF) that allows systems to indicate the product codes to be excluded when placing content in a format. Examples: Auto-Car, Fast Food and Toys. This element should only be included on elements that are Breaks.

BxfMessage/BxfData/Format/FormatStructure/FormatElements/PrivateInformation/FederalSource: The FederalSource extension element is a configuration type (CONF) that allows systems to indicate the source of the content. Examples: Live, Network, Recorded. This element should only be included on elements that are Segments.

BxfMessage/BxfData/Format/FormatStructure/FormatElements/PrivateInformation/InsertAfterElementId: The InsertAfterElementId extension element shall be used to associate a format element to the element that proceeds it. If it not present or blank on a format add message then it shall be the first element in the format. The InsertAfterElementId shall be included in add, change, remove and query reply messages.

BxfMessage/BxfData/Format/PrivateInformation/EiCode: This is the "Target Age Range" and shall be used for (for example) FCC children's programming reporting requirements.

BxfMessage/BxfData/Format/PrivateInformation/ParentalRating: These rating values are for the format. More than one parental rating may be sent in a message. Attributes and elements of Parental Ratings are:

- ***/ProgramContent/ParentalRating/@region:** The region contains the mapped region identifier (CONF).
- ***/ProgramContent/ParentalRating/Rating/@dimension:** The dimension contains the mapped dimension value (CONF). Examples: TV Parental.
- ***/ProgramContent/ParentalRating/Rating/@value:** The value contains the mapped value for the dimension (CONF). Examples: TV-PG, TV-G, R.
- ***/ProgramContent/ParentalRating/ContentAdvisory:** The "ContentAdvisory" element contains the mapped content advisory value (CONF). It includes an attribute of language. Examples: Dialogue, Some Sexual Situations, Harsh Language.

12.2 Requesting Format Information (Queries) (Normative)

The following sections identify how an application can request the format of another application. Each section is annotated, indicating if the functionality is recommended, optional or to be avoided.

12.2.1 Obtaining All Format Information

The conventions for handling BXF queries for format information are described in the following sections. Recommend limited query capabilities.

12.21.1 Request Message

Applications serving as a repository of format information shall provide the capability to consume a request for all of their format values.

Applications may provide the capability to create a request message to get all of another application's formats, as well as the functionality to consume the corresponding reply.

The example message contains a request for all format values:

Format– Get All

Recommended Conventions – Format Query Request

BxfMessage/@ext:usage: The usage shall be "Format Query".

BxfMessage/@messageType: The message type shall be "Request".

BxfQuery/Where: For the Format Query Get All Request, the <WhereClause> element for "Format="*" shall be supported, which indicates that all format values are being requested.

12.2.1.2 Reply Message

Applications shall provide the capability to reply to a format query request even if the response is 'not_supported'. This can be achieved through a reply that contains the content data or through a reply that references one or more file locations where the content data resides. For example:

Format – Get All Reply (with data)
Format – Get All Reply (single file)

Recommended Conventions – Format Query Reply

BxfMessage/@ext:usage: The usage shall be "Format Query Reply".

BxfMessage/@messageType: The message type shall be "Reply".

BxfMessage/BxfQueryResponse: Format query response capabilities should be implemented. If this capability is not provided by an application, then the "not_supported" error type should be utilized on acknowledgement messages.

BxfMessage/BxfQueryResponse: The reply message for a "get format" request may utilize the file-based approach (described below) or chunking if the message size exceeds 2 MB.

Additional conventions are specified in Section 6.5, Reply Messages.

12.2.2 Obtaining Type-Specific Format Values

The conventions for handling BXF queries for type-specific content information are described in the following sections. At least one of the recommended request messages is required.

12.2.2.1 Request Message

Applications shall provide the capability to consume a request for type-specific content values. Applications should support a query by GUID.

The following example shows the query requests that are supported:

Format – Get by Format ID (GUID)

Recommended Conventions – Type-Specific Format Query Request

BxfMessage/@ext:usage: The usage shall be “Format Query”.

BxfMessage/@messageType: The message type shall be “Request”.

BxfMessage/@messageType/Format/FormatId: A format-related GUID.

The “=” operator and the “and” operator shall be supported; all other operators are optional.

12.2.2.2 Reply Message

Applications should provide the capability to reply to a “type-specific” configuration query request. For example:

Format – Get by Format ID (GUID) Reply

Recommended Conventions – Type-Specific Format Query Reply

A type-specific format query reply should contain all of the associated format information that is associated with the request. For example, “Get by ID” should return format and format elements if it exists and is applicable to the requested type.

BxfMessage/@ext:usage: The usage shall be “Format Query Reply”.

BxfMessage/@messageType: The message type shall be “Reply”.

Additional conventions are specified in **Reply Messages**.

12.2.3 Obtaining Format Values through Advanced Queries

The BXF protocol supports advanced querying capabilities and includes conditional logic (e.g., and/or logic) and operators (e.g., <, >, =, etc.). Applications are not required to provide support for advanced operations other than the get type-specific examples that were described in the previous section.

In the context of format, these capabilities should typically be avoided since they will require significant time and effort to develop.

12.3 Notifying Others when Formats are Added

12.3.1 Format

The following example shows a format Add:

Format – Add

Recommended Conventions – Add Format

BxfMessage/@ext:usage: The usage shall be “Format”.

BxfMessage/@action: The action shall be “add”.

BxfMessage/@messageType: The message type shall be “Information”.

12.4 Notifying Others when Formats are Updated

12.4.1 Format

The following example shows a format Update:

Format – Update Format Elements

Recommended Conventions – Update Format

BxfMessage/@ext:usage: The usage shall be “Format”.

BxfMessage/@action: The action shall be “update”.

BxfMessage/@messageType: The message type shall be “Information”.

12.5 Notifying Others when Formats are Removed

12.5.1 Format

The following example shows a format removal:

Format – Remove Format

Recommended Conventions – Remove Format Element

BxfMessage/@ext:usage: The usage shall be “Format”.

BxfMessage/@action: The “action” shall be “remove”.

BxfMessage/@messageType: The message type shall be “Information”.

13 Schedule

13.1 Overview (Informative)

The Schedule element within the BXF protocol allows applications to share schedule-related information such as playlist and as-run information.

Note: Scheduling-related capabilities are applicable to scheduling aware applications such as program management, traffic and automation applications.

13.1.1 Schedule Conventions (Normative)

Recommended Conventions - Schedule

The following conventions apply to all schedule messages:

BxfMessage/BxfData/Schedule: The Schedule element shall be utilized to notify other applications of changes to schedule information.

BxfMessage/BxfData/Schedule/@scheduleStart: The Schedule shall use the canonical representation of UTC (e.g., 2007-02-01T15:30:00.0Z) with no time zone offset. As a result, all remaining date and time values within the message are also implicitly using canonical UTC.

BxfMessage/BxfData/Schedule/@scheduleEnd: The Schedule shall use the canonical representation of UTC (e.g., 2007-02-01T15:30:00.0Z) with no time zone offset. As a result, all remaining date and time values within the message are also implicitly using canonical UTC.

BxfMessage/BxfData/Schedule/ScheduleEvent/EventData/@eventType: The recommended usage of eventType is as follows:

- Primary: The Primary event type shall be used when identifying program and non-program primary events.
- NonPrimary: The NonPrimary event type shall be used when identifing non-program, non-primary (secondary) events.
- Primary-ProgramHeader: The Primary-ProgramHeader event type should be used when identifying the beginning of a program. While a sample message in SMPTE 2021 shows the SegmentNumber set to 1 in the case of a Program Header, all sample messages in SMPTE 2021 are informative in nature. To avoid confusion on the part of systems implementing SMPTE 2021, it is strongly recommended in the case of Program Headers that the SegmentNumber shall be set to 0, as a ProgramHeader is not a segment.
- Primary-BreakHeader: The Primary-BreakHeader event type should be used when identifying the beginning of a break.
- Comment: The Comment event type shall be used when identifying a non-switchable event other than Primary-ProgramHeader or Primary-BreakHeader.
- Macro: The Macro event type shall be used when identifying a collection of events or commands stored as a Macro event.

BxfMessage/BxfData/Schedule/ScheduleEvent/EventData/EventId/EventId:
BxfMessage/BxfData/Schedule/AsRun/CompleteAsRun/EventData/EventId/EventId:
BxfMessage/BxfData/Schedule/AsRun/BasicAsRun/AsRunEventId/EventId: The value in the element contains a GUID value that uniquely identifies each event. Schedule message creators and consumers should determine (1) if these identifiers need to be persisted internally within their application and (2) how they will be persisted. If the events will be referenced in future messages (e.g., as-run that is based on received playlist), then the event identifiers shall be persisted so that they can be added to the as-run message. Similarly, if the message consumer needs to correlate events in a message with internal application events, then the identifiers should be persisted. The implementation approach will vary by application, but it typically involves adding a mapping capability or custom properties.
Channel conventions:

BxfMessage/BxfData/Schedule/Channel/Name, and/or
BxfMessage/BxfData/Schedule/Channel/@channelNumber: The Channel Name element and/or channelNumber is used to uniquely identify the channel. At least one of these two Channel identifiers shall be included in a schedule message.

BxfMessage/BxfData/Schedule/Channel/Description: The Channel Description element should be included for informational purposes.

13.1.2 Event Time and Duration (Normative)

The BXF protocol provides a variety of approaches for representing event time and duration. The details are the responsibility of the implementer. The following sections identify recommendations for representing an event's start date/time and duration. The primary objective is to explicitly specify what the value should contain and, where applicable, how the value was calculated.

13.1.2.1 Start Date/Time

***/StartDateTime:** SmpteTimeCode should be used for StartDateTime since this element provides the capability to identify the starting time at the frame (or even sub-frame) level. Receiving systems shall support both SmpteTimeCode and UtcDateTime formats.

***/StartDateTime/SmpteTimeCode:** The time given is a relative offset based on the date specified, so the value shall be in UTC. Per the SMPTE 12M-1 standard that is the basis for this field, the time code value's syntax indicates if it is a drop frame or a non-drop frame time code. For example, a value of 15:30:30:20 indicates that an event starts 30 seconds and 20 frames past 3:30 pm UTC, and that this is not a drop frame time code.

If the message originator does not support frames, then the time code value should be passed with zero frames (e.g., 15:30:00:00. If this is the case, the NominalFlag (StartDateTime/@nominalFlag) attribute shall be included with a value of true.

Sub-frame level values are not supported and shall be ignored.

If an application utilizes milliseconds internally, care should be taken to ensure that frames are not lost during the conversion process.

This approach allows the message consumer to (1) determine if the event start time contains frame-level precision and (2) determine if the value accounts for drop frames so that the event can be processed correctly.

13.1.2.2 Duration

The event duration is provided in the following elements:

BxfMessage/BxfData/Schedule/ScheduledEvent/EventData/LengthOption/Duration:
BxfMessage/BxfData/Schedule/AsRun/CompleteAsRun/EventData/LengthOption/Duration:
BxfMessage/BxfData/Schedule/AsRun/BasicAsRun/AsRunDetail/LengthOption/Duration:
SmpteDuration should be used for Duration since this element provides the capability to identify the starting time at the frame (or even sub-frame) level. The receiving system shall support both formats.

***/Duration/SmpteTimeCode:** The duration shall be passed in the element. Per the SMPTE 12M-1 standard that is the basis for this field, the time code value's syntax indicates if it is a drop frame or a non-drop frame time code.

If the message originator provides limited support for frames (e.g., duration is stored in a manner that allows the frame count to be determined, but drop frame algorithms are not supported), then the duration value should be passed with frames and the syntax should indicate that this is not a drop frame value (e.g., 15:30:00:10, where the colon indicates that this is a non-drop frame time code). If the application supports drop frame time codes, then the duration value should reflect this (e.g., 15:30:00;10, where the ";" indicates that this is a drop frame time code).

If the message originator does not support frames, then the duration value may be passed without frames (e.g., 15:30:00) and the NominalFlag element shall be included with a value of true. In this case, the message consumer shall consider the event duration an approximation. For systems supporting frame rates greater than 30 frames per second, time shall be expressed in UTC, as the SMPTETimeCode field supports values from 0 to 29.

Regardless of the time type used, if the nominal flag (Duration/@nominalFlag) is set to true, times shall be seen as approximate and the receiving system may modify the times as it sees fit; if the nominal flag is false or not set, times shall be considered to be absolute and will be used by the receiving system as a direct instruction.

13.1.3 Content in Schedule Messages (Normative)

This information pertains to the usage of the Content element within schedule messages. The Content element is specified at the following locations in the schedule message:

- **BxfMessage/BxfData/Schedule/ScheduleEvent/Content/**
- **BxfMessage/BxfData/Schedule/ScheduleEvent/ScheduleElements/Content/**
- **BxfMessage/BxfData/Schedule/AsRun/CompleteAsRun/Content/**
- **BxfMessage/BxfData/AsRun/CompleteAsRun/ScheduleElements/Content/**
- **BxfMessage/BxfData/Schedule/AsRun/BasicAsRun/Content/**

Note: Content in schedules is represented differently than in other BxfMessages as it does not have distinction of Program Content vs. Non-Program content and the various elements that go with those types.

Content information is expressed in two primary contexts, as explicit content information and as usage of content.

It can be observed that content (i.e., type ContentMetaData), is used in most /BxfData message capacities at some level. This may seem obvious, but it is important to note that most messaging data is concerned with content in some form or another. The differences appear in the context of the reference. In this structure, there is provision to provide details concerning media for the material information of the content.

Content references in a schedule item shall assume a context that is best defined by the schedule item event information. For example, a primary event may use a video clip as part of the primary 'play-to-air' content, while a non-primary (e.g. secondary) event may use a static image to perform a branding over-lay of the primary video using a separate content reference locally paired with the non-primary event element usage.

Thus, it is by virtue of using BXF in the context of playlist scheduling that systems ensure references to /Content shall always be paired with /EventData.

13.1.3.1 Use of MediaLocation Elements

The MediaLocation element is included within the Content as follows:

- ***/Schedule/ScheduledEvent/Content/Media/MediaLocation/Location**
- ***/Schedule/ScheduleElements/Content/Media/MediaLocation/Location**
- ***/Schedule/AsRunSchedule/CompleteAsRun/Content/Media/MediaLocation/Location**
- ***/Schedule/AsRunSchedule/BasicAsRun/Content/Media/MediaLocation/Location**

It is important to note in the BxfSchema that this type is used to define the element /Media, which is declared as optional and unbounded in occurrence. As such, multiple references for /Media may be declared for a given /ContentId. This allows a system to provide more information concerning the location and usage contexts for the content. This ability allows a system to specify both source and usage information about content associated with a schedule item.

It makes it possible for a system to declare a primary and backup source for a schedule item either as separate usage references or separate media instances. It also allows for the possibility for data to be passed for the purpose of ingest of content metadata concerning discreet instances of material.

The choice elements specified in BXF for /Location, defined by the LocationType, provide a mechanism for declaring information about either a unique instance or unique usage of content in a schedule item.

Recurrent usage of the /Media element allows that both instance and usage information can be passed in the schedule data. So one instance of /Media may include location information about the source to play from, while another instance provides information about the source media for the content representation.

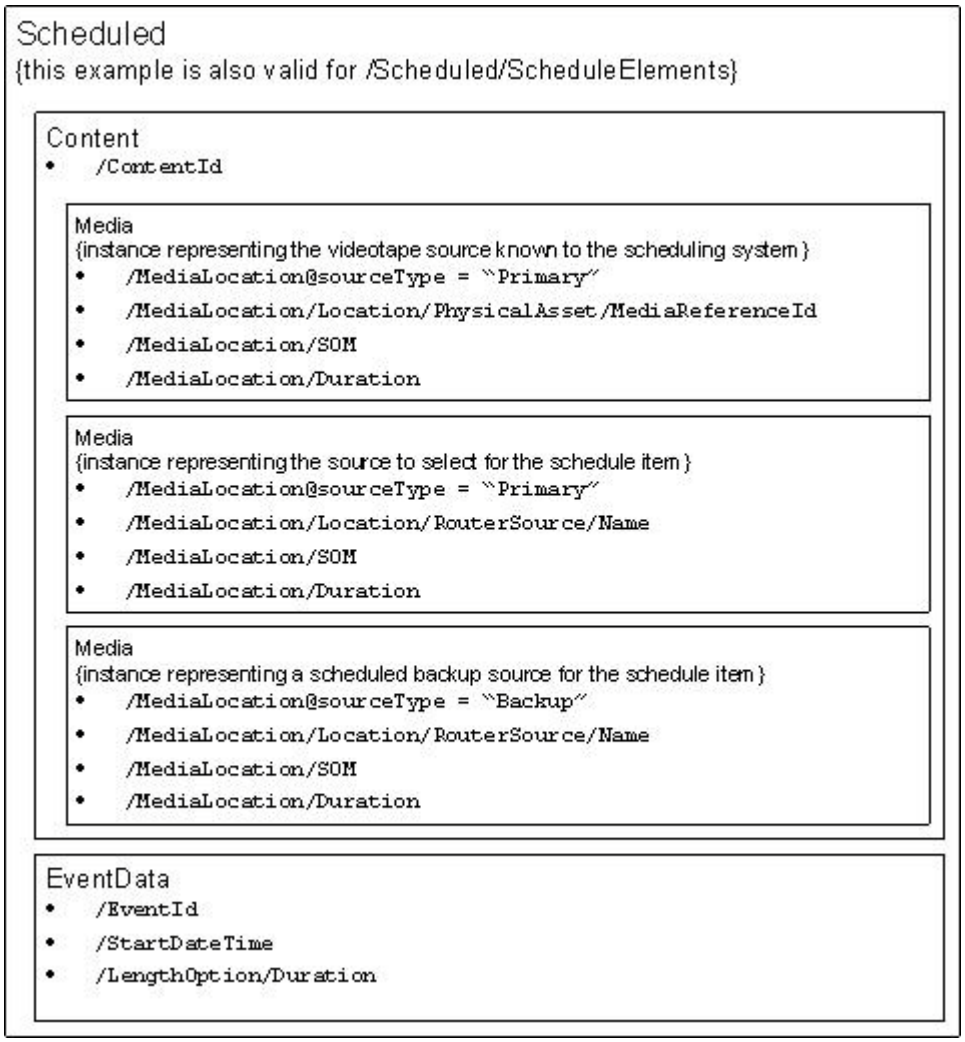


Figure 1 – Example Usage Structure

13.1.3.1.1 Material Usage

***/Media/MediaLocation/Location/RouterSource:** The router source element provides a choice element whereby a source may be explicitly or implicitly identified to the automation system for the given schedule item and the associated content.

***/Media/MediaLocation/Location/RouterSource:** The presence of a "RouterSource" in an instance of "Media" used in a playlist context shall always refer to the usage of the content identified by the relevant "ContentId". This means that the timing information for this instance of "Media" shall be interpreted as the play

from and duration information for the schedule item relative to the source timing data of the given content identified by “ContentId”.

***/Media/Location/SOM:** It is important to note that the timing data for /Media/Location/MediaLocation/RouterSource shall always be explicit timing data relative to the source media. In the case of a live source, 00:00:00;00 should be passed, as the time is not likely to be known. In the case of pre-recorded sources, this shall be the source timecode index.

***/Media/Location/Duration:** It is important to note that the timing data for /Media/Location/MediaLocation/RouterSource shall always be explicit timing data relative to the source media. In the case of a live source, 00:00:00;00 should be passed, as the time is not likely to be known. In the case of pre-recorded sources, this shall be the source timecode index.

In practice there is a usage concept here that requires agreement between two systems operating data exchange with this model, whereby both systems can recognize the values supplied and their relative meaning.

For instance, it is uncommon for a scheduling system to be aware of discreet source names for a playout schedule. In contemporary schedule interchange models – prior to BXF – the process of identification for the requisite discreet playout sources is either the burden of the automation system or a bit of schedule import logic for processing the schedule data supplied by the scheduling system.

13.1.3.1.1.1 Implied Source

It is important that the scheduling and automation systems concerned have agreed to employ references that are meaningful to either both systems or to one system and some transformation logic bound in between them.

For example, a scheduling system provides playlist information with reference to source only for live events. In this case, the reference is a source name derived from the real source (e.g. “Studio-A”, “NET-7”, etc.). These may or may not directly correspond to discreet switcher sources. For pre-recorded items, it may only declare a default value such as “Local” or “Default” for the source information.

In the playout environment, business rules concerning the non-live sources may include a normal practice to attempt to play from a video server source. In this case, the operational staff can deal with exceptions where the video server clip and/or the player source are unavailable.

In another example, the scheduling system may operate in a slightly enhanced capacity to recognize that the playout environment is based on usage of a mixture of player sources. This array may include the course elements of Live, Server and Tape sources.

In this example, the scheduling system could then build a playlist with schedule items referring to primary playout source and preferred alternatives or ‘protection’ sources. In this way, a /RouterSource/Name could be set to “Server”, and a backup assignment – using another instance of /Media for the same /Content – made using /RouterSource/Name set to “Tape”.

13.1.3.1.1.2 Scheduled Source

A scheduled source shall be explicitly named in the /RouterSource/Name value. The value will correspond to a configured source that the relevant automation system supports for control of player and switcher devices.

In this case, there is no business logic needed in between the sender and receiver of the message as the values are known to both in a directly useful manner.

13.1.3.1.1.3 Scheduled Backup Source

A backup source shall be identified as a scheduled source that is explicitly declared as a backup source type in the attributes of the /Location element:

/Media/Location@sourceType = “Backup”

This instance of /Media may be passed within the same /Content instance as that of the primary content, or it may be passed in a separate /Scheduled (or /Scheduled/ScheduleElements) instance. The former case is logical to use if the /ContentId for the “Backup” is the same as the “Primary”. The latter is useful for updating but also for scheduling a “Backup” with a different /ContentId.

It should be noted that there is no explicit prohibition to scheduling multiple “Backup” sources. Support for such facility is, however, entirely dependent on the capabilities of the relevant scheduling and playout system elements communicating using this model.

13.1.3.1.2 Declaring Source Media

***/Media/MediaLocation/Location:** There are four choice elements that provide source media information in a schedule item:

/Location/PhysicalAsset

/Location/AssetServer

/Location/RouterSource

/Location/Satellite

With the exception of /Satellite, the choice elements available for /MediaLocation all provide the ability to specify information about the pre-recorded material. Satellite data is useful for expressing the availability and tuning schedule for the Satellite source, which is, in effect, the source media for a live source. In all of these cases, the timing information associated, /MediaLocation/Location/SOM and /MediaLocation/Location/Duration, refer to the source timing of the content for the given /ContentId.

13.1.3.1.2.1 PhysicalAsset

***/Media/MediaLocation/Location/PhysicalAsset:** This choice element provides a mechanism for supplying the source media information in the playlist data.

The most immediately obvious application of the /PhysicalAsset element is identification of a removable media (e.g., a videotape).

13.1.3.1.2.2 AssetServer

***/Media/MediaLocation/Location/AssetServer:** This choice element provides a mechanism for supplying the source media information about a material clip instance in a managed volume location such as a video file server.

13.1.3.2.3 RouterSource

***/Media/MediaLocation/Location/RouterSource:** This choice element provides a mechanism for supplying the source media information about a material clip instance from one location to another (for example from a News Room, News Truck or Satellite to a playout or Record device).

13.1.3.1.2.4 Satellite

***/Media/MediaLocation/Location/AssetServer:** This choice element provides a mechanism for supplying the source media information about a Satellite feed that may be routed as a source.

13.2 Requesting Schedule Information (Queries) (Normative)

BxfMessage/BxfQuery: This element may be implemented, but this is not required. If these capabilities are not provided by an application, then the "not_supported" error type shall be utilized on acknowledgement messages.

BxfMessage/BxfQueryResponse: This element may be implemented, but this is not required. If these capabilities are not provided by an application, then the "not_supported" error type shall be utilized on acknowledgement messages.

13.2.1 Requesting Schedule Information (Normative)

Schedule-aware applications shall provide the capability to consume a request for their schedule information based on the types of schedules they support (e.g., as-run).

Applications may provide the capability to create a request message to get another application's schedule information as well as the functionality to consume the corresponding reply.

Get schedule example messages:

AsRun (Basic) – Get Full Day by Channel
AsRun (Basic) – Get by Channel and Date Range
EPG Schedule – Get by Channel and Date Range
Playlist – Get by Channel and Date/Time Range

Recommended Conventions – Get Schedule Information

BxfMessage/@ext:usage: The usage shall be "Playlist Query".

BxfMessage/@messageType: The message type shall be "Request".

The minimum recommended query is by channel and date/time range.

Note: To query for a complete day, specify the start and end of the day in the date/time range.

13.2.2 Replying to Schedule Requests

Get as-run reply example message:

AsRun (Basic) – Get Reply
EPG Schedule – Get Reply

Recommended Conventions – Get Schedule Reply

BxfMessage/@ext:usage: The usage shall be "Playlist Query Reply".

BxfMessage/@messageType: The message type shall be "Reply".

BxfMessage/@errors: If the requested query functionality for schedule request query is not supported by an application, then the associated acknowledgement should include an "error" status and an error type of "not_supported".

Additional conventions are specified in **Reply Messages**.

13.3 EPG Schedule Notifications (Normative)

Recommended Conventions – EPG Schedule

BxfMessage/@usage: The usage shall be “EPG Schedule”.

BxfMessage/BxfData/Schedule/ScheduledEvent/ScheduleElements/EventData/@eventType: EPG schedules shall only contain “Primary-ProgramHeader” type events.

BxfMessage/BxfData/Schedule/ScheduledEvent/Content: EPG schedules should contain content data (when available) as this is the mechanism for instructing the EPG system about an event.

BxfMessage/BxfData/Schedule/ScheduledEvent/EventData/PrimaryEvent/ProgramEvent/SegmentNumber: The EPG schedule event segment number shall be set to “0” since segment information is not applicable.

13.3.1 Add EPG Schedule

Add EPG schedule example message:

EPG Schedule - Add

13.4 Playlist Notifications (Normative)

BXF definition (Playout schedule):

> *“A list of events to be played in sequential order based on the instructions included as part of the schedule. This includes which content should be played and any special effects that should be added to or aired in conjunction with the content.”*

The playlist message supports the following capabilities:

- Notification capability for add, change and remove of scheduled events. Note that these messages can originate from any participating application, unlike the traditional model where playlist information is sent from traffic to automation.
- Support for a primary schedule and alternate schedules.
- Support for primary and non-primary (e.g., secondary) events.
- Support for program and non-program (e.g., commercials) content.
- Support for different types of schedules (e.g., sales, program, EPG, etc.).
- Support for one or more events in a message, providing the capability to update a single event or multiple events at the same time.

The playlist message does not support the following capabilities:

- Inline recording of programming for later playback – The Record Order is intended for this purpose, refer to the Content Transfer section for additional information.

Recommended Conventions - Playlist

BxfMessage/@ext:usage: The usage shall be “Playlist”.

BxfMessage/BxfData/Schedule/@type: When sending a Playlist, the “type” attribute is used to determine if a playlist message is for a primary or alternate schedule.

BxfMessage/BxfData/Schedule/@type: For playlist, the “type” attribute is used to determine if the schedule pertains to a primary or alternate schedule.

BxfMessage/BxfData/Schedule/ScheduleName: If the schedule is an alternate then this shall be used to indicate which alternate schedule to use. On an alternate schedule, an identifier unique within the channel and day shall be provided.

BxfMessage/BxfData/Schedule/@scheduleId: The “scheduleId” attribute identifies the schedule.

BxfMessage/BxfData/Schedule/@dayPattern: The “dayPattern” attribute may be included to identify the broadcast day in a schedule.

BxfMessage/BxfData/Schedule/@scheduleStart: The “scheduleStart” attribute may be included to represent the start of the broadcast day. If this attribute is included, the scheduleEnd attribute should also be included.

BxfMessage/BxfData/Schedule/@scheduleEnd: The “scheduleEnd” attribute may be included to represent the end of the broadcast day. If this attribute is included, the scheduleStart attribute should also be included.

BxfMessage/BxfData/Schedule/@invokeSchedule: The “invokeSchedule” capabilities are not recommended for Playlist messages.

BxfMessage/BxfData/Schedule/Channel: A single playlist message shall only contain events for an individual channel.

BxfMessage/BxfData/Schedule/ScheduledEvent: Message creators may send a schedule for non-contiguous events or for contiguous events (e.g., sending an entire schedule).

- A schedule may contain program and non-program information, as well as primary and secondary events.
- If a message contains multiple events, then they shall be listed in chronological order, regardless of whether they are contiguous or non-contiguous.

BxfMessage/BxfData/Schedule/ScheduledEvent/EventData/InsertAfterEventId: This element is used to identify the preceding event. This value shall be sent for all events other than the first event in a schedule. The existence of this element shall take precedence over the event’s start date/time for placement purposes. If this is not provided then events shall be processed based solely on start date/time.

13.4.1 Add Playlist

Add playlist example message:

Playlist - Add

Add messages shall define events for air time that is not currently scheduled for the specified channel, air date and schedule.

BxfMessage/@messageType: The message type shall be “Information”.

BxfMessage/BxfData/@action: The action shall be “add”.

13.4.2 Update Playlist

Update playlist example messages:

Playlist – Update Spot
Playlist – Replace Spot
Playlist – Replace Program

Recommended Conventions – Update Playlist

Update messages shall manipulate events for air time that is currently scheduled for the specified channel, air date and schedule.

BxfMessage/@messageType: The message type shall be “Information”.

BxfMessage/BxfData/@action: If the originating application updated playlist events within a schedule, then this attribute shall be set to “update”. If the message contains a combination of event additions, updates and removals, then the action attribute shall be set at the appropriate level to reflect these changes.

13.4.3 Remove Playlist

Remove playlist example message:

Playlist – Remove (remove program segment)

Recommended Conventions – Remove Playlist

Remove messages shall apply to the specified channel, air date, schedule and/or time range.

BxfMessage/@messageType: The message type shall be “Information”.

BxfMessage/BxfData/@action: The action shall be “remove”.

13.5 As-Run Notifications (Normative)

BXF definition:

> *“A term typically applying to broadcast playout schedules referring to the exact events that aired during a specific period of time, usually a broadcast day. The "as-run" includes the exact airtimes and durations of each event as well as the status of how each event actually aired.”*

The as-run message supports the following capabilities/approaches:

- As-run messages will only contain as-run event-related additions and updates.
- Notification capability for as-run events that are generated directly after air out. These messages typically originate from an automation application. Receiving applications should process the message containing the as-run events immediately upon receipt. The notification supports for the following:
    - One or more events in a single as-run message
    - Additions or updates in a single message

Recommended Conventions – As-run

The following conventions apply to all as-run messages:

BxfMessage/@ext:usage: The usage shall be “AsRun”.

BxfMessage/BxfData/Schedule: For As-run, a single as-run message shall not not contain multiple channels (i.e., multiple elements with different channel values).

BxfMessage/BxfData/Schedules/AsRun: Message creators may send an as-run using the “CompleteAsRun” or the “BasicAsRun” element depending on the capabilities of their application. Message consumers shall provide functionality to handle messages that contain as-run information in both of these formats.

- BxfMessage/BxfData/Schedule/AsRun/CompleteAsRun element usage shall be limited to a single event since the AsRunDetail information relates to a single event.

- BasicAsRun element may be used to send multiple events within a single message. Non-primary event information should be included with associated primary events. Each event shall be within a separate BxfMessage/BxfData/Schedules/AsRun element.

BxfMessage/BxfData/Schedule/AsRun/CompleteAsRun/EventData/PrimaryEvent/NonProgramEvent/Details/SpotType: New comment events that are added to an as-run shall use the primary, non-program event structure with the spot type set to that appropriate configuration value (e.g., Comment).

BxfMessage/BxfData/Schedule/AsRun/CompleteAsRun/EventData/PrimaryEvent/ProgramEvent/SegmentNumber: A single segment program event shall have a segment number of “1”.

All executed (i.e., ran to air) events that are specified in the corresponding playlist shall be included in the as-run information. For example, if a primary header event or a break header event is included in the playlist then it will be provided in corresponding as-run message(s) if the event was executed.

BxfMessage/BxfData/@action: The remove action shall not be supported within as-run messages. Messages will typically contain add actions, but updates may be provided to certain elements (e.g., comments)

13.5.1 Add As-run

Add as-run example message:

AsRun (Complete) – Add
AsRun (Basic) - Add

Recommended Conventions – Add As-run

BxfMessage/@messageType: The message type shall be “Information”

BxfMessage/BxfData/@action: For Add AsRun schedule the message should contain new as-run information (“add” action), but it may also contain updates.

13.5.2 Update As-run

Update as-run example message:

AsRun (Complete) – Add Comment
AsRun (Basic) – Update
AsRun (Basic) – Add Comment

Recommended Conventions – As-run Changes

BxfMessage/@messageType: The message type shall be “Information”

BxfMessage/BxfData/@action: For AsRun changes to the schedule the message should contain new as-run information (“update” action), but it may also contain adds.

13.5.3 Remove As-run

BxfMessage/BxfData/@action: For AsRun messages the “action” of “Remove” shall not be supported.

Annex A Supported Extensions (Normative)

A.1 Overview

The following sections describe shared extensions to the BXF protocol. These extensions cover functionality that is not currently defined in the protocol but strongly recommended for interoperability.

Shared extensions to the BXF protocol utilize the following namespace:

`http://smpte-ra.org/schemas/2021/2011/BXF/`**`Extension`**

A.2 Usage Attribute

The “usage” extension attribute provides a logical definition of the payload of the message so that message consumer can process it without having to infer or interpret the intent of the message. Each usage has a single definition, as defined in Annex B, Message Type Usages.

Applications shall include a usage attribute within their message.

A.3 Filename Element

The “Filename” extension element provides the name and location of a file that contains a BXF message. Note that the name of the file shall comply with the file naming convention that is identified in the BXF protocol documentation.

Applications should include a Filename element in Information and Reply messages where the amount of data is extensive and messaging-based delivery is not desired.

A.4 Root Element Extensions

Certain types within BXF are commonly returned in query response messages. These types cannot be validated against an XML schema because they are not public elements. Extensions are provided to redefine the following types as public elements: Configuration, Content, ContentTransfer, Format and Schedule.

A.5 Extension Schemas

The following links contain the extension schemas.

Extension Schema	Description
bxfschema-extension.xsd	The entry point to all BXF extensions
bxtypes-extension.xsd	Defines common elements, types and attributes and are utilized within the BXF extensions
configuration-extension.xsd	Defines extensions to the BXF configuration element
content-extension.xsd	Defines extensions to the BXF content element
contentmetadata-extension.xsd	Defines extensions to the BXF content metadata element or its sub-elements
contenttransfer-extension.xsd	Defines extensions to the BXF content transfer element
element-extension.xsd	Defines extensions to the BXF element
format-extension.xsd	Defines extensions to the BXF format element
schedule-extension.xsd	Defines extensions to the BXF schedule element

Annex B Message Type Usages (Normative)

Message type usages are provided for the following reasons:

- **Prevent Message Type Collisions** – Ensures that the same message type usage value is not used by different applications in different contexts with different meanings.

- **Ensures a Single Definition** – Each message type usage has a single purpose that is identified in its description below.

Each message type usage consists of a "base" usage type. The name should intuitively describe the information being sent in one to four words (e.g., Playlist).

B.1 Registered Message Type Usages

The following message type usages are registered. Their definition indicates their intended purpose.

B.1.1 Configuration Message Type Usages

Configuration message type usages shall be used to share and synchronize runtime configuration information. Configuration messages utilize the Configuration element located in configuration.xsd.

Name	Description	Category
Reference Data Query	A request for all reference data types and values within an application.	Request
Reference Data Query Reply	A reply containing an application's reference data types and values.	Reply
Reference Data	A notification of reference data changes including updates and removes.	Information
Reference Data Reply	A reply containing an application's reference data values for a request.	Reply
Reference Data Request	A request to change reference data that requires a reply (e.g., add genre).	Request

B.1.2 Content Message Type Usages

Content message type usages shall be used to exchange content information between applications. Content messages utilize the Content and ContentMetadata elements located in content.xsd and contentmetadata.xsd, respectively.

Name	Description	Category
Content	A notification of content changes including adds, updates and removes.	Information
Content Query	An application-initiated request for content information.	Request
Content Query Reply	A reply containing an application's content values.	Reply
Media	A notification of media changes including adds, updates and removes where there is not associated content,	Information
Media Query	An application-initiated request for media information.	Request
Media Query Reply	A reply containing an application's media values.	Reply
Program Content Header	A notification containing program content header information, including the program identifier, usage and rating information. It is typically published by a program management system to a traffic system	Information
Program Content Header Query	An request for program header information. It is typically published by a traffic system to a program management system	Request
Program Content Header Query Reply	A reply containing an application's program header values.	Reply

B.1.3 Content Transfer Message Type Usages

Content transfer message type usages shall be used to transfer content information and/or essence between applications. Content Transfer messages utilize the ContentTransfer element defined in contenttransfer.xsd.

Name	Description	Category
Dub List	A notification to dub material. It is typically published by a program management or a traffic system to an automation or DAM system.	Information
File Transfer	A notification to transfer material. It is typically published by a program management or a traffic system to an automation or DAM system.	Information
Purge List	A notification to purge material. It is typically published by a program management or a traffic system to an automation or DAM system.	Information
Record List	A notification to record material. It is typically published by a program management or a traffic system to an automation or DAM system.	Information

B.1.4 Diagnostic Message Type Usages

Diagnostic message type usages shall be used to monitor the health of the connected applications. Diagnostic messages utilize the BxfMessage element defined in bxfschema.xsd.

Name	Description	Category
Application Acknowledgement	Application acknowledgement that a message has been processed.	Acknowledgement
Application Heartbeat	Application heartbeat request from one application to other application(s).	Heartbeat

B.1.5 Format Message Type Usages

Format message type usages shall be used to identify changes to format information. Format messages utilize the Format element defined in format.xsd.

Name	Description	Category
Format	A notification of format changes including adds, updates and removes.	Information
Format Query	An application-initiated request for format information.	Request
Format Query Reply	A reply containing an application's format values.	Reply

B.1.6 Schedule Message Type Usages

Schedule message type usages shall be used to identify changes to schedule information. Schedule messages utilize the Schedule element defined in schedule.xsd.

Name	Description	Category
AsRun	A notification of actual playout information including events that were added, updated and/or removed. It is typically published by an automation system to a traffic, sales and/or program management system.	Information
AsRun Query	A request for an as-run schedule generally for a specific date/time/channel range.	Request
AsRun Query Reply	A reply to the AsRun Query.	Reply
EPG Schedule	A notification of EPG schedule changes including adds, updates and removes. It is typically published by a program management system to an EPG system.	Information
EPG Schedule Query	A request for an EPG schedule generally for a specific date/time/channel range.	Request
EPG Schedule Query Reply	A reply to the EPG Schedule Query.	Reply
Playlist	A notification of playlist changes including adds, updates and removes. It is typically published by traffic or program management systems to automation systems.	Information
Playlist Query	A request for a playlist schedule generally for a specific date/time/channel range.	Request
Playlist Query Reply	A reply to the Playlist Query.	Reply
Sales Schedule	A notification of sales schedule changes (i.e., spots and breaks) including adds, updates and removes. It is typically published by a sales system to a traffic and/or program management system.	Information
Sales Schedule Query	A request for a sales schedule generally for a specific date/time/channel range.	Request
Sales Schedule Query Reply	A reply to the Sales Schedule Query.	Reply
Program Schedule	A notification of program schedule changes including adds, updates and removes. It is typically published by a program management system to a traffic and/or sales system.	Information
Program Schedule Query	A request for a program schedule generally for a specific date/time/channel range.	Request
Program Schedule Query Reply	A reply to the Program Schedule Query.	Reply

B.2 Information Message Syntax

Information messages contain a request to do something or a notification that something happened, neither of which warrants a reply. Message type usage syntax:

[baseMessageType]	*example*s: Content, Playlist, AsRun

B.3 Request and Reply Message Syntax

Request messages contain a request to do something that warrants a reply. Reply messages are generated in response to requests. Message type usage syntax:

[baseMessageType] **Request**	*example*: Content Request
[baseMessageType] **Reply**	*example*: Content Reply

B.4 Query Message Syntax

Query for message types are a special type of request that identifies the message as a query as opposed to a request to do something (e.g., add a new value). Query for messages always include a reply. Message type usage syntax:

[baseMessageType] **Query**	*example*: Content Query
[baseMessageType] **Query Reply**	*example*: Content Query Reply

B.5 Custom Message Type Usages

Custom message type usages utilize the usage syntax that is described in the previous sections; however, a custom identifier is included to indicate that the message is not for public consumption.

Applications may need to create custom usage types that vary from a registered message type usages in one or more ways (e.g., data format is unique due to customer requirements). Message type usage syntax:

[baseMessageType] **(identifier)**	*example*: Content (ABC Client)
[baseMessageType] Request **(identifier)**	*example*: Content Request (ABC Client)
[baseMessageType] Reply **(identifier)**	*example*: Content Reply (ABC Client)
[baseMessageType] Query **(identifier)**	*example*: Content Query (ABC Client)
[baseMessageType] Query Reply **(identifier)**	*example*: Content Query Reply (ABC Client)

Note: Applications are responsible for preventing collisions when custom message types are utilized.

B.6 Application-Specific Message Type Usages

Application-specific message type usages utilize the message type usage syntax that is described in the previous sections; however, an “application” designation is included to indicate that the message is not for public consumption.

Applications may need to create message type usages that are specific to the application itself and are not intended for general consumption. Message type usage syntax:

[baseMessageType] **(application)**	*example*: Content (Some App)

[baseMessageType] Request **(application)**	*example*: Content Request (Some App)
[baseMessageType] Reply **(application)**	*example*: Content Reply (Some App)
[baseMessageType] Query **(application)**	*example*: Content Query (Some App)
[baseMessageType] Query Reply **(application)**	*example*: Content Query Reply (Some App)

Note: Applications are responsible for preventing collisions when application-specific message types are utilized.

Annex C Error Handling (Informative)

C.1 Overview

The BXF protocol provides a robust solution for identifying errors in messages. It is critical that the message consumer validate the structural and data aspects of the message to ensure it is valid before processing it. It is also possible that the message is valid, but the consumer cannot process it due to resource availability or internal (logic) errors. It is critical that all of these conditions be accurately represented in the associated acknowledgement or reply message that indicates that the received message was rejected.

A brief overview of the BXF error handling capabilities is provided below. Refer to the BXF documentation for additional information.

BxfMessage/@status: Identifies the overall status of the message including invalid, valid, error, warning and OK.

BxfMessage/@error: Identifies errors using the BxfMessage element. One or more errors may be specified. Errors should be set on this attribute if it is not possible to set errors contextually. This represents the default location for errors.

BxfMessage/@errorDescription: Describes errors using the BxfMessage element. One or more error descriptions may be specified. Error descriptions should be set on this attribute if it is not possible to set error descriptions contextually. This represents the default location for error descriptions.

C.2 Error Types

Error Pattern	Description	Examples
element_does_not_exist	When a message is evaluated by the receiving system and a specific value is requested that does not exist in the other system, the reply message would indicate that using this error. As a recommended practice, implementers are encouraged to use a suffix string to communicate the name of the requested element.	element_does_not_exist:*Name*
system_unavailable	When the system processing the message is unable to complete an internal operation such as connecting to database or other system failures, then this error should be returned assuming that the ability to return a message is still possible. As a recommended practice, implementers are encouraged to use a *suffix* string to communicate the name of the system or the component that became *unavailable* during the processing of the message.	System_unavailable*:ProgramLibrary* System_unavailable*:1002*
not_supported	When a request is made to a system to perform a query or other update function which that system does not support, then this error message should be returned. As a recommended practice, implementers are encouraged to use a suffix string to communicate the name of the *not supported* service.	not_supported*:adhocQuery*

Error Pattern	Description	Examples
duplicate_message	When the receiving system identifies the message as a duplicate of a previously received message that has been processed successfully, this error should be sent. A suffix in this case is not necessary	duplicate_message
processing_exception	When the receiving system fails in any way to return the proper data or update a data record as requested by the other system, this error should be sent. A recommended practice, implementers are encouraged to use a detailed description as to the cause of the exception in the @errorDescription attribute. A suffix is completely optional	processing_exception processing_exception*:1001*
*_out_of_range	When a message provides data for an element or a query and the values provided are out of the valid range for the object, this error should be sent. As recommended practice, the name of the object that is out of range is appended as a prefix replacing the asterisk "*".	*StartDateTime*_out_of_range
*_missing	When required data values are missing from the message, this error should be sent. As recommended practice, the name of the object that is associated with the missing data values is appended as a prefix replacing the asterisk "*".	*ParentalRating*_missing
*_change_denied	When the system processing the message refuses the add, update or remove due to security issues related to the user requesting or any other restriction, then the object name being changed is substituted for the "*" with this error.	*ContentId*_change_denied

C.3 Setting Errors Contextually

Most elements in BXF contain optional attributes for the error type and description. These errors are contextual and should pertain to the element, sub-element or attribute that they are associated with.

***/@error:** It is critical that the error identify the type of error(s).

***/@errorDescription:** It is critical that the error description provide a detailed description of the error(s).

C.4 Identifying Errors and Warnings

If a message contains errors but the consumer can still process it successfully (e.g., a description is too long so it is truncated), then the appropriate "error" attribute should be set indicating that the "*value*_out_of_range" and the "messageStatus" should be "warning".

All structural errors (i.e., message format errors) that are identified by the message consumer should be specified in a contextual manner by adding error and/or errorDescription attributes to the offending elements, or their parent elements.

All data errors that are identified by the consumer may be specified in a contextual manner, identifying the offending element and/or attribute values.

If a consumer cannot identify an error contextually, then the Bxfmessage@error and/or BxfMessage@errorDescription attributes should be set by default indicating why the message is being rejected.

Use "|" as the delimiter on the errorDescription attribute since an error message may contain the colon character. The following error examples demonstrate multiple error types (delimited with a space) and descriptions (delimited with a bar) on a single attribute:

> error="element_does_not_exist:*Name ContentId*_change_denied"
>
> errorDescription="Alternate ID "*Name*" is required and was not provided**|**ContentId element has an action of update, but updates are not supported"

C.5 Errors Implementation within BXF

The following constructs are defined within the BXF protocol for error handling:

```
<xs:simpleType name="BxfElementaryErrorExt">
      <xs:annotation>
            <xs:documentation>Type for an elementary
            error</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:string">
                  <xs:pattern
                  value="(system_unavailable|not_supported|duplicate_messag
                  e|processing_exception)(:.*)?"/>
      </xs:restriction>
</xs:simpleType>

<xs:simpleType name="BxfElementaryError">
      <xs:annotation>
            <xs:documentation>Type for an elementary
            error</xs:documentation>
      </xs:annotation>
            <xs:union memberTypes="pmcp:elementaryErrorType
      BxfElementaryErrorExt"/>
</xs:simpleType>
<xs:simpleType name="BxfError">
      <xs:list itemType="BxfElementaryError"/>
</xs:simpleType>
```

The following construct is defined within the PMCP protocol for error handling:

```
<xsd:simpleType name="elementaryErrorType">
      <xsd:annotation>
            <xsd:documentation>Type for an elementary
            error</xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
            <xsd:pattern
            value="(element_does_not_exist|.*_out_of_range|.*_missing|.*_c
            hange_denied)(:.*)?"/>
            </xsd:restriction>
</xsd:simpleType>
```

Figure C.1 – PMCP Errors

BXF error attribute is defined as a list of tokens in order to allow reporting of multiple error values separated by spaces. An error value is a string containing no spaces and conforming to the patterns defined in the BXF schema or the error patterns inherited from the PMCP schema (Figure C.1). Implementers can extend BXF and PMCP error patterns by using an optional suffix string separated by a colon (:). Similarly, implementers can extend certain error patterns by using a prefix string. Prefix and Suffix strings are arbitrary strings containing no spaces. As recommended practices implementers are encouraged to use a standard set of values when possible as shown in the list below. In addition, as recommended practice implementers are encouraged to use of the @errorDescription attribute to include explanatory description for added readability.

Annex D Protocol Transport (Informative)

D.1 Legend

The following legend applies to the diagrams in the request, information and heartbeat message sections below:

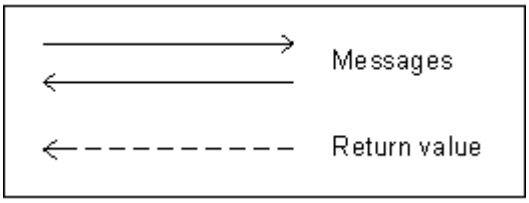


Figure D.1

D.2 Request Messages

Request messages represent a request for information (i.e., query) or a request to do something that warrants a reply (e.g., add content and return unique identifier).

D.2.1 Normal Flow – Acknowledge Using Return Value

An event [1] within Application-A results in a connection from Application-A (client) to Application-B (server). Application-A sends a request [2.1] to Application-B for processing. Application-B ensures that the request is valid per the schemas [2.2] and provides an acknowledgement (status=valid) in the return value [2.3]. Application-B processes the request and generates a reply [3]. Application-B (client) connects to Application-A (server) and sends the reply, completing the request [4].

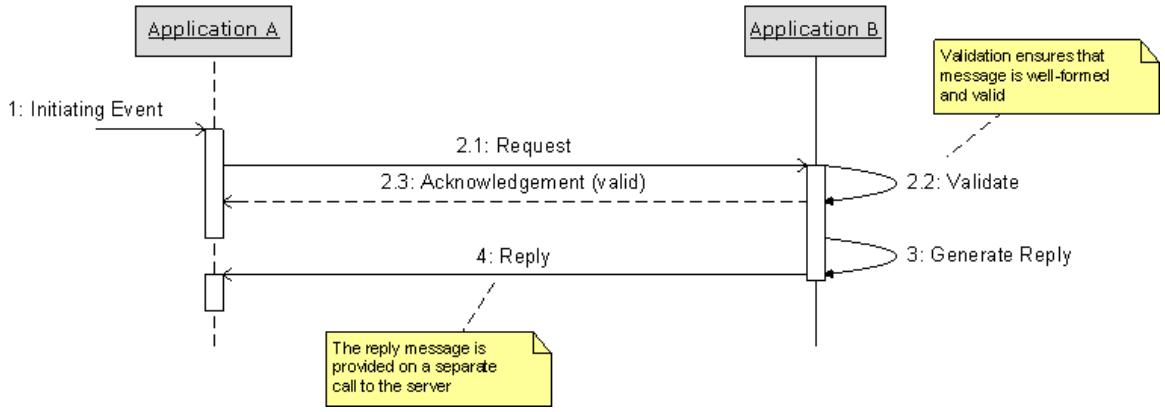


Figure D.2

The following flow is similar to the one above. The only difference is that Application-B performs additional validations and provides an acknowledgement (status=OK) in return value [2.3].

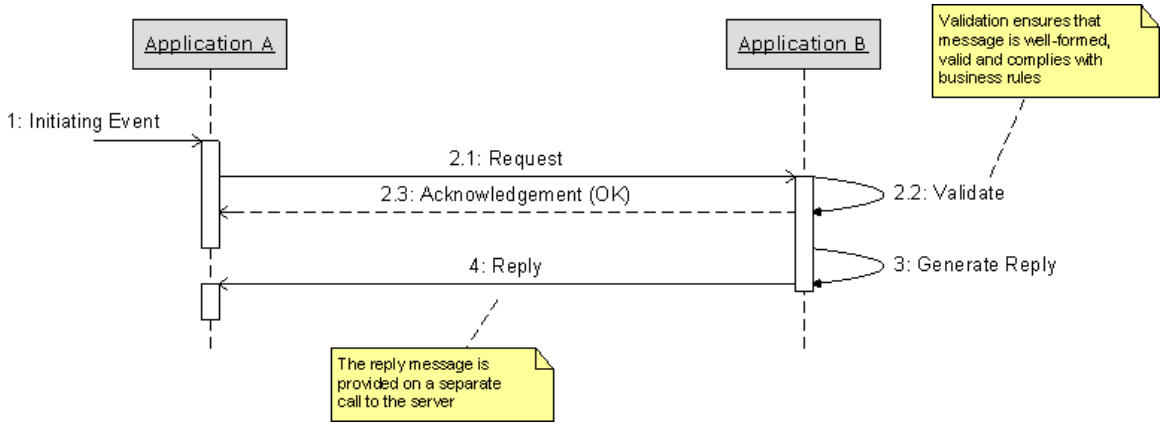


Figure D.3

D.2.2 Normal Flow – Acknowledge Using Message

An event [1] within Application-A results in a connection from Application-A (client) to Application-B (server). Application-A sends a request [2] to Application-B for processing. The Application-B saves the request, staging it for processing — no validation occurs and an acknowledgement is not provided in the return value. Application-B validates the request [3], generates an acknowledgement (status=valid) [4], creates a client and sends the acknowledgement to Application-A (server) [5]. Application-B processes the request and generates a reply [6]. Application-B (client) connects to Application-A (server) and sends the reply, completing the request [7].

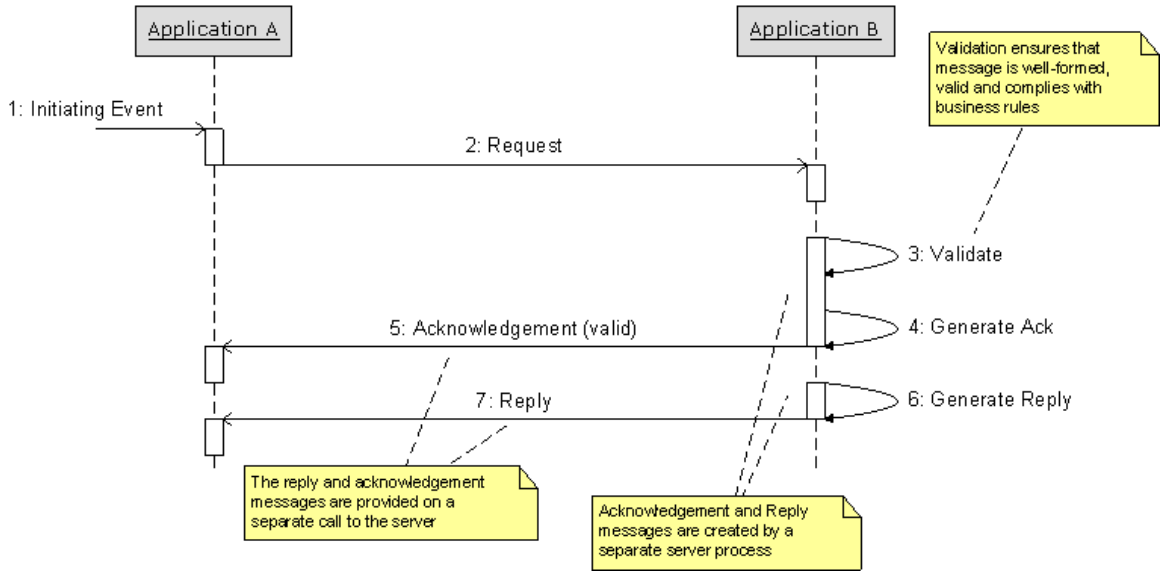


Figure D.4

D.2.3 Error – Acknowledge Using Return Value

An event [1] within Application-A results in a connection from Application-A (client) to Application-B (server). Application-A sends a request [2.1] to Application-B for processing. Application-B identifies one or more validation errors [2.2] and acknowledges (status=invalid) the message in the return value [2.3].

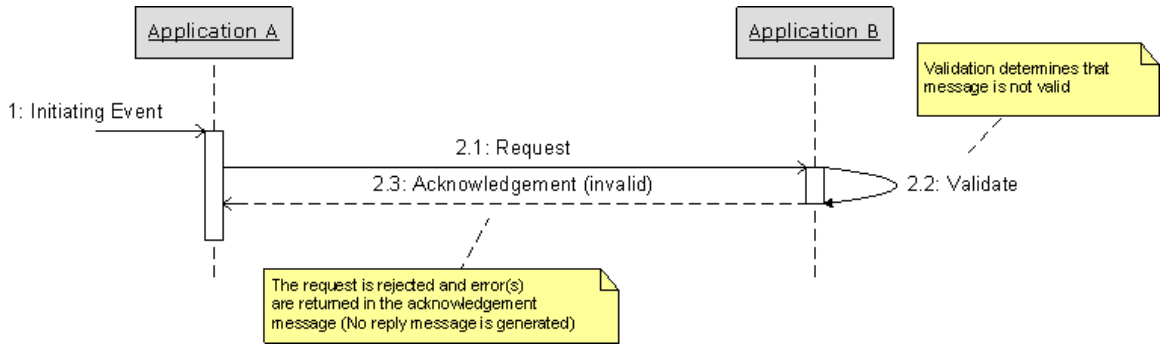


Figure D.5

The following flow is similar to the one above. The only difference is that Application-B identifies one or more processing errors [2.2] and acknowledges (status=error) the message in the return value [2.3].

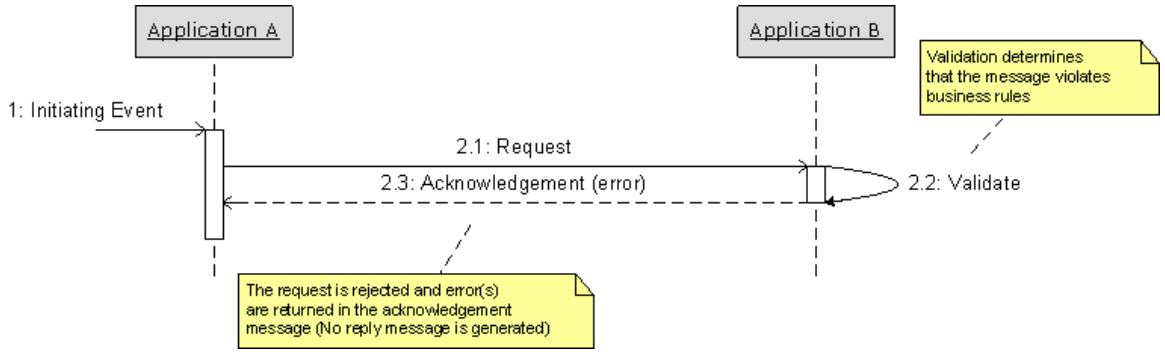


Figure D.6

D.2.4 Error – Acknowledge Using Message

An event [1] within Application-A results in a connection from Application-A (client) to Application-B (server). Application-A sends a request [2] to Application-B for processing. The Application-B saves the request, staging it for processing — no validation occurs and an acknowledgement is not provided in the return value. Application-B identifies one or more processing errors [3], generates an acknowledgement (status=error), creates a client and sends the acknowledgement to Application-A (server) [4].

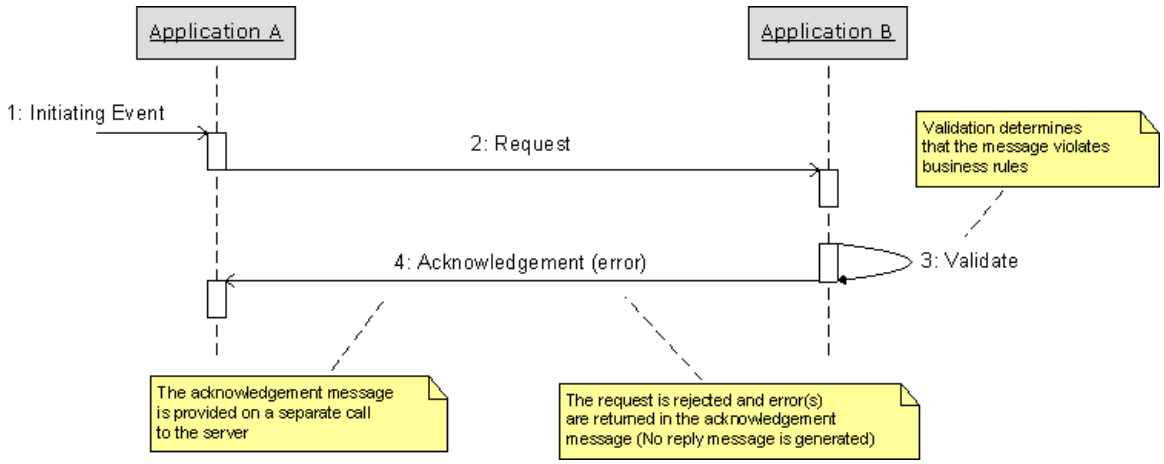


Figure D.7

D.3 Information Messages

Information messages represent a request to do something that does not warrant a reply (e.g., update description on content).

D.3.1 Normal Flow – Acknowledge Using Return Value

An event [1] within Application-A results in a connection from Application-A (client) to Application-B (server). Application-A sends an information/notification message [2.1] to Application-B for processing. Application-B ensures that the message is valid per the schemas [2.2] and provides an acknowledgement (status=valid) in the return value [2.3]. Application-B processes the message [3].

Note: This approach should use error handling as defined in section Error – Multiple Acknowledgements.

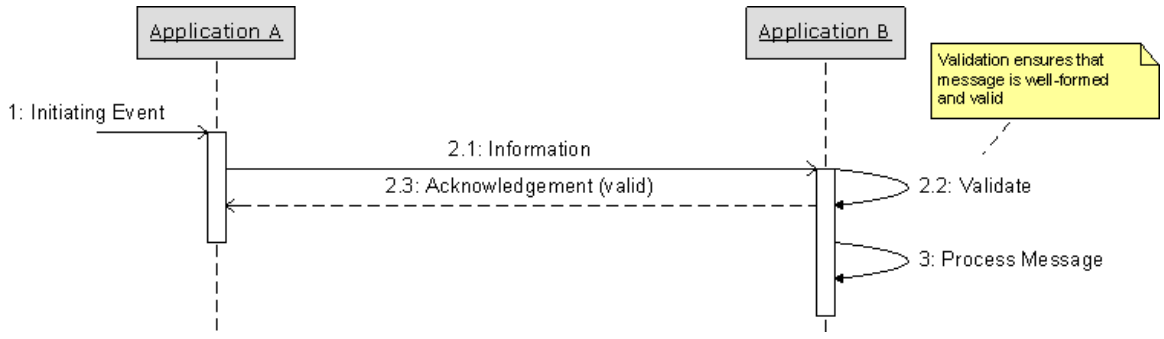


Figure D.8

The following flow is similar to the one above. The only difference is that Application-B processes the message [2.3] and provides an acknowledgement (status=OK) in the return value [2.4].

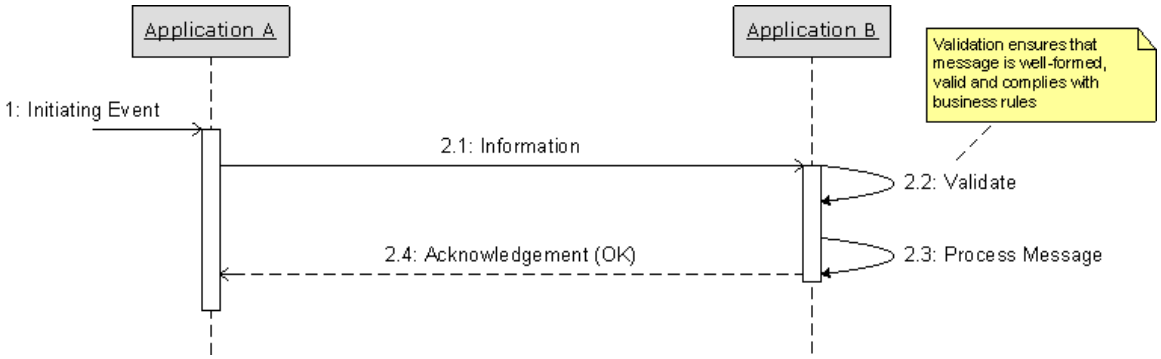


Figure D.9

D.3.2 Normal Flow – Acknowledge Using Message

An event [1] within Application-A results in a connection from Application-A (client) to Application-B (server). Application-A sends an information/notification message [2] to Application-B for processing. The Application-B saves the message, staging it for processing — no validation occurs and an acknowledgement is not provided in the return value. Application-B validates the message [3], processes it and generates an acknowledgement (status=OK) [4], creates a client and sends the acknowledgement to Application-A (server) [5].

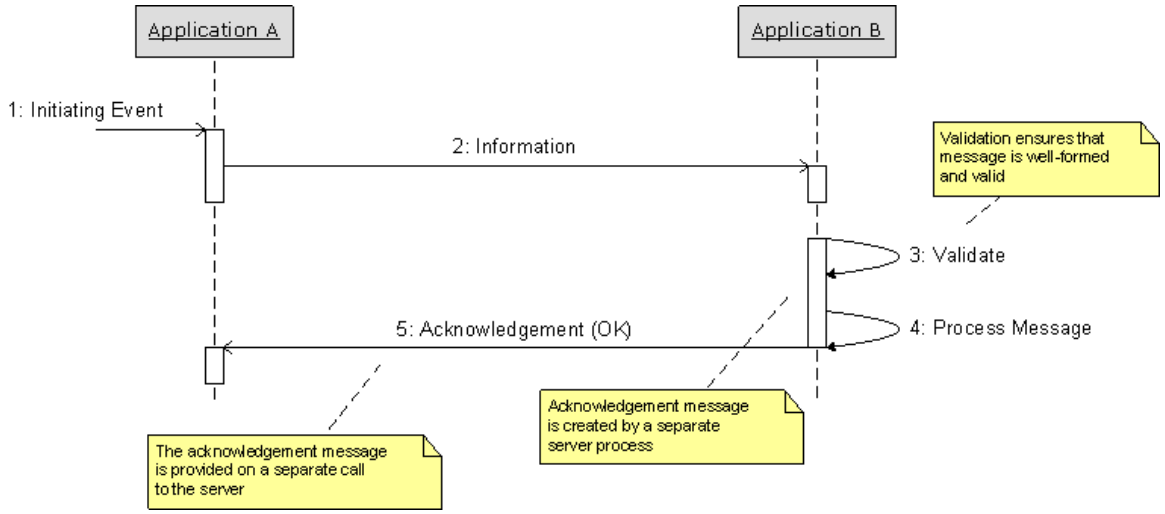


Figure D.10

D.3.3 Error – Acknowledge Using Return Value

An event [1] within Application-A results in a connection from Application-A (client) to Application-B (server). Application-A sends an information/notification message [2.1] to Application-B for processing. Application-B identifies one or more validation errors [2.2] and acknowledges (status=invalid) the message in the return value [2.3].

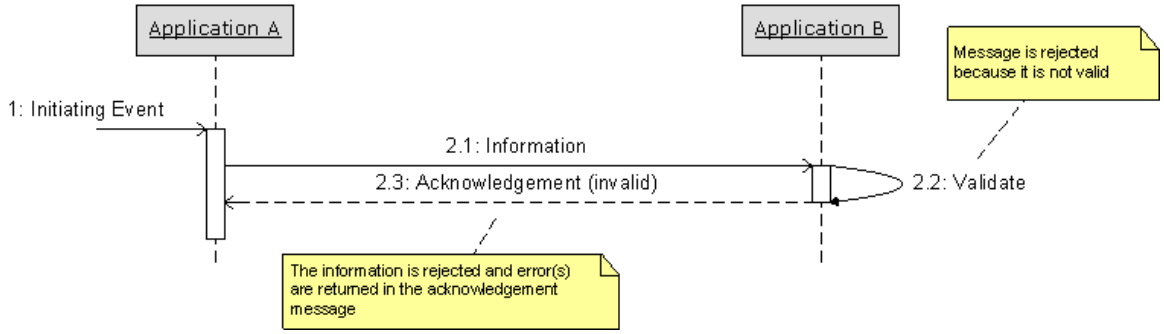


Figure D.11

The following flow is similar to the one above. The only difference is that Application-B identifies one or more processing errors [2.2] and acknowledges (status=error) the message in the return value [2.3].

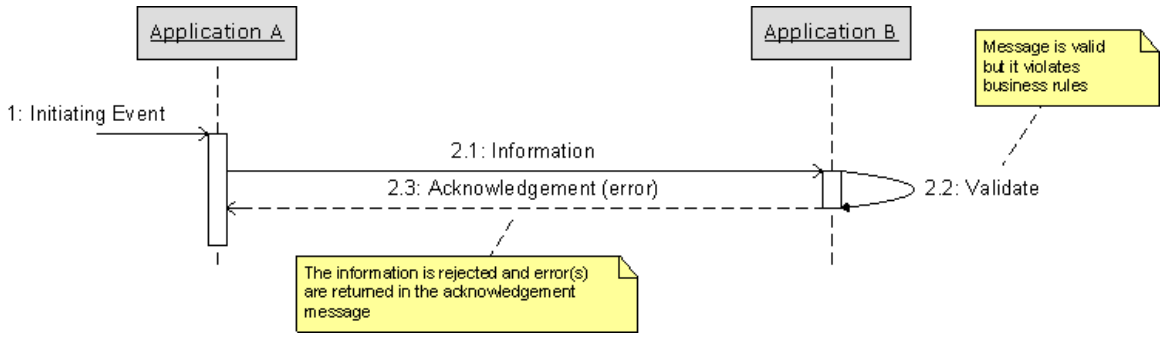


Figure D.12

D.3.4 Error - Acknowledge Using Message

An event [1] within Application-A results in a connection from Application-A (client) to Application-B (server). Application-A sends an information/notification message [2] to Application-B for processing. The Application-B saves the message, staging it for processing — no validation occurs and an acknowledgement is not provided in the return value. Application-B identifies one or more processing errors [3], generates an acknowledgement (status=error), creates a client and sends the acknowledgement to Application-A (server) [4].

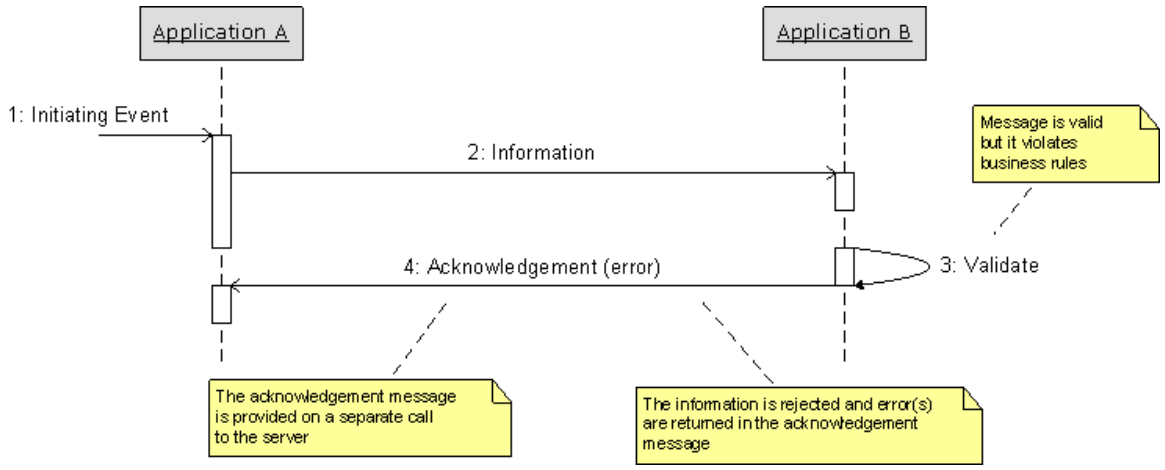


Figure D.13

D.3.5 Error – Multiple Acknowledgements

An event [1] within Application-A results in a connection from Application-A (client) to Application-B (server). Application-A sends an information/notification message [2.1] to Application-B for processing. Application-B ensures that the message is valid per the schemas [2.2] and provides an acknowledgement (status=valid) in the return value [2.3]. Application-B identifies one or more processing errors [3], generates an acknowledgement (status=error), creates a client and sends the acknowledgement to Application-A (server) [4].

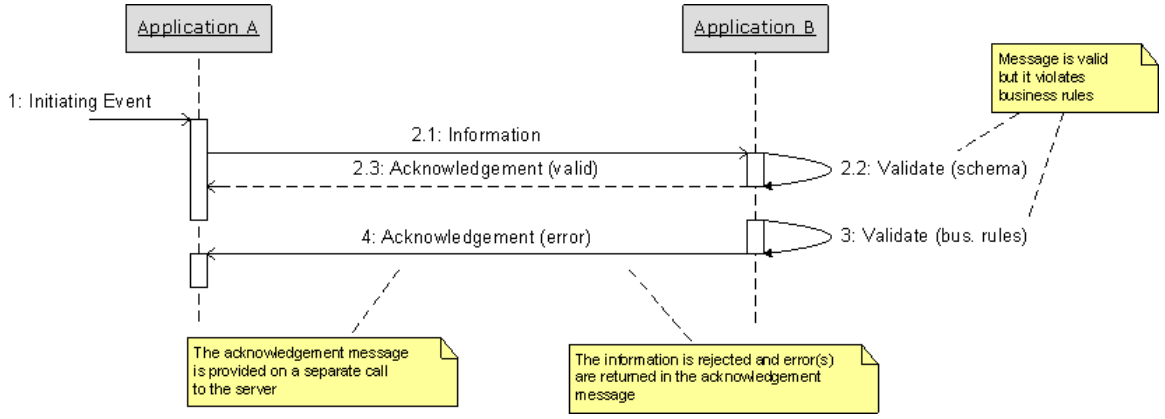


Figure D.14

D.4 Heartbeat Messages

Heartbeat messages represent a connectivity “health check”. Heartbeat messages should test the bi-directional nature of the application-to-application exchange.

D.4.1 Normal Flow – Acknowledge Using Return Value

An event [1] within Application-A results in a connection from Application-A (client) to Application-B (server). Application-A sends a heartbeat message [2.1] to Application-B for processing. Application-B ensures that the heartbeat is valid per the schemas [2.2] and provides an acknowledgement (status=OK) in the return value [2.3].

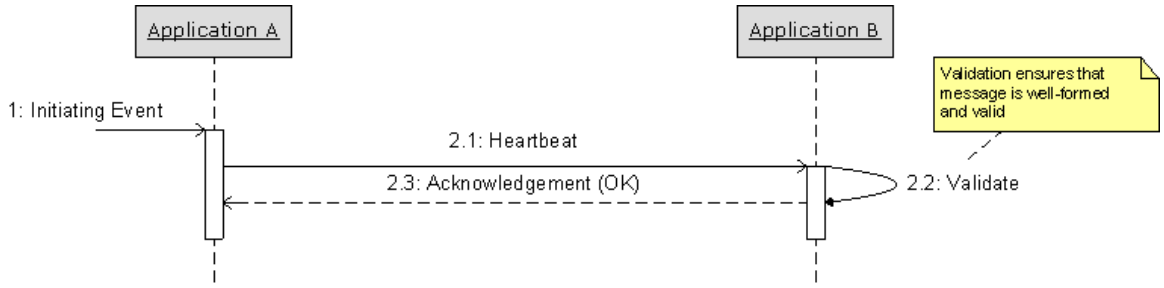


Figure D.15

D.4.2 Normal Flow – Acknowledge Using Message

An event [1] within Application-A results in a connection from Application-A (client) to Application-B (server). Application-A sends a heartbeat message [2] to Application-B for processing. The Application-B saves the heartbeat, staging it for processing — no validation occurs and an acknowledgement is not provided in the return value. Application-B validates the heartbeat [3], generates an acknowledgement (status=OK), creates a client and sends the acknowledgement to Application-A (server) [4].

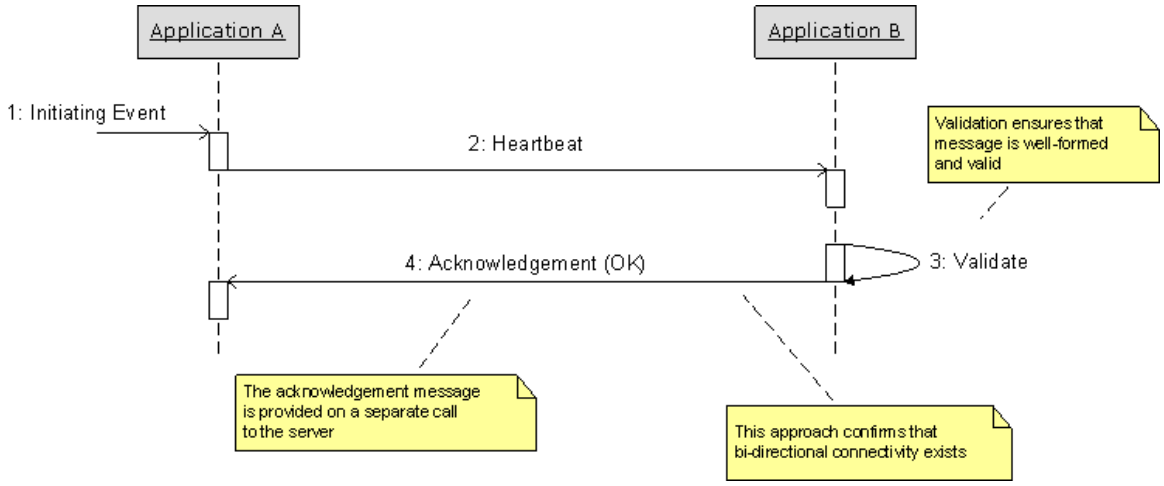


Figure D.16

D.4.3 Error – Acknowledge Using Return Value

An event [1] within Application-A results in a connection from Application-A (client) to Application-B (server). Application-A sends a heartbeat message [2.1] to Application-B for processing. Application-B identifies one or more validation errors [2.2] and acknowledges (status=invalid) the message in the return value [2.3].

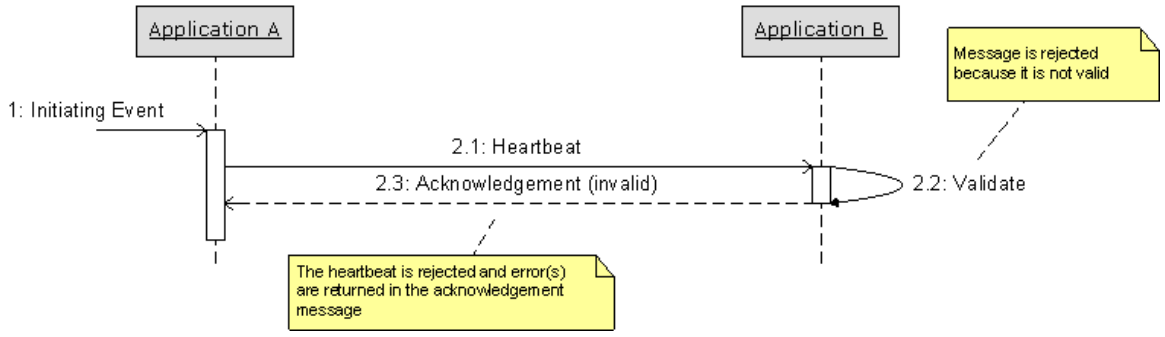


Figure D.17

The following flow is similar to the one above. The only difference is that Application-B identifies one or more processing errors [2.2] and acknowledges (status=error) the message in the return value [2.3].

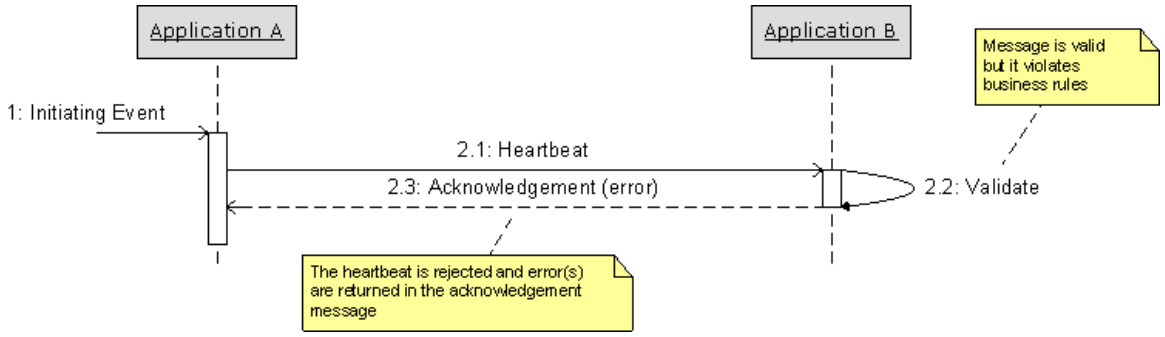


Figure D.18

D.4.4 Error – Acknowledge Using Message

An event [1] within Application-A results in a connection from Application-A (client) to Application-B (server). Application-A sends a heartbeat [2] to Application-B for processing. The Application-B saves the heartbeat, staging it for processing — no validation occurs and an acknowledgement is not provided in the return value. Application-B identifies one or more processing errors [3], generates an acknowledgement (status=error), creates a client and sends the acknowledgement to Application-A (server) [4].

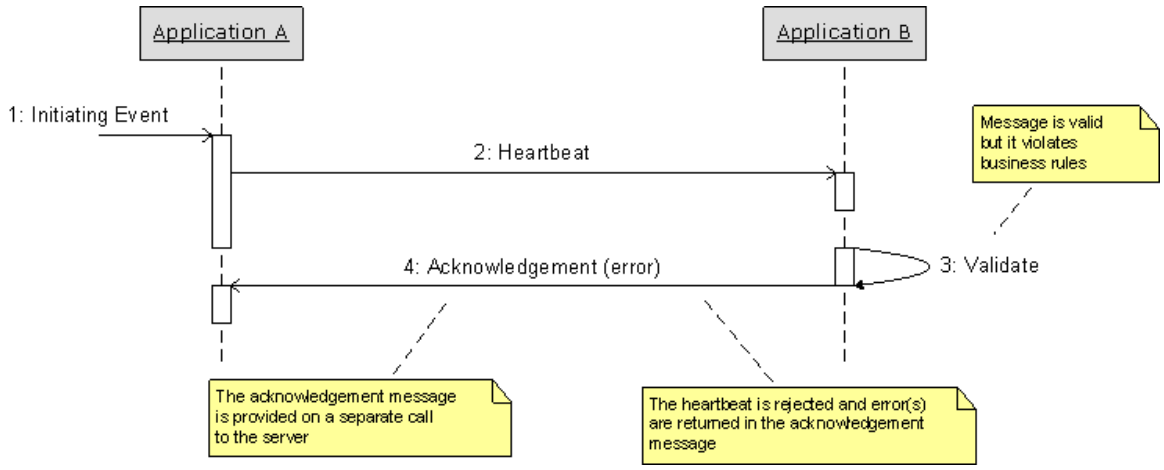

Application A
Application B
1: Initiating Event
2: Heartbeat
Message is valid
but it violates
business rules
4: Acknowledgement (error)
3: Validate
The acknowledgement message
is provided on a separate call
to the server
The heartbeat is rejected and error(s)
are returned in the acknowledgement
message

Figure D.19

Annex E Bibliography (Informative)

W3C Extensible Markup Language (XML) 1.0 (Fourth Edition), http://www.w3.org/TR/xml/#sec-well-formed