WebSphere MQ Service Definition

Bindings and WSDL Extensions

Version 1.0

November 2007

Mark Phillips
m8philli@uk.ibm.com

Matthew Golby-Kirk
mgk@uk.ibm.com

Notices

Table of Contents

1 Introduction

1.1 Background

This document and the related WebSphere® MQ (WMQ) IRI Specification document ([WMQ-IRI]) specify how to describe WMQ applications as services, and how to send SOAP messages over WebSphere MQ in a standardised way.

The specifications define both SOAP and non-SOAP WMQ bindings, and so can describe traditional WMQ applications as services, as well as SOAP services which use WMQ as a transport.

The main purposes of the these two documents are:

- To allow WMQ applications to be described and catalogued (for instance in a services registry);
- To describe the way in which applications interact with a WMQ service;
- To facilitate the development of tools in support of the above.

The specification will also be of interest to providers of intermediary services such as an Enterprise Service Bus, providing for example a SOAP/HTTP to WMQ transport switch. This specification does not detail how to design or configure such an intermediary, but using the service definition will help to ensure proper interoperation with WMQ applications and services.

The documents cover five major areas.

1) The IRI specification which defines identifiers for the messaging resources used by WMQ services. Note that the IRI is fully specified in a separate document ([WMQ-IRI]).
2) The properties used by a WMQ service definition to provide more information about a WMQ service. Properties include information about how to connect to a WMQ infrastructure; the queues and/or topics the service uses; the service's message exchange pattern; and the qualities of service the service expects (for example, in terms of message priority, and persistence).
3) The way to construct and interpret the SOAP and non-SOAP WMQ messages which communicate with a service.
4) A set of examples illustrating the combinations of IRI's and service invocation messages.
5) A WSDL binding that MAY be used to describe WMQ services

1.2 Out of Scope

It *would* be possible to define WMQ bindings for many of the WS-* standards (e.g. WS-Addressing, or WS-Security) by specifying how to carry the relevant headers and attributes in native (non-SOAP) WMQ messages – for example in MQRFH2 folders. This specification DOES NOT attempt to define any such native WMQ binding for WS-* standards.

1.3 SOAP Binding Context

A binding is specified for both SOAP 1.1 and SOAP 1.2 using the SOAP 1.2 Protocol Binding Framework to describe how SOAP binds to WebSphere MQ.

This specification is modelled on three of the binding specifications that have been created for SOAP 1.2. These are the HTTP binding which is described in [SOAP 1.2 Part 2], the SOAP Email binding which is described in [SOAP Version 1.2 Email Binding], and the [SOAP-JMS] binding.

1.4 Standard Prefix Mappings

The following table lists the standard prefix mappings which we assume to hold throughout this specification: Properties (both those defined in this specification and those defined in other specifications) are named with XML qualified names (QNames). Property values are determined by the Schema type of the property, as defined in the specification which introduces the property.

Prefix	Namespace	Notes
wmqservice	http://www.ibm.com/xmlns/prod/wmq/bindings/1.0	This specification
context	http://www.w3.org/2003/05/soap/bindingFramework/ExchangeContext/	SOAP 1.2 MEP context
fail	http://www.w3.org/2003/05/soap/bindingFramework/ExchangeContext/FailureReason	SOAP 1.2 MEP failure reasons
mep	http://www.w3.org/2003/05/soap/mep/	SOAP 1.2 MEP
oneway	http://www.w3.org/2006/08/soap/mep/one-way/	SOAP 1.2 one-way MEP
xsd	http://www.w3.org/2001/XMLSchema	XML Schema
wsdl	http://schemas.xmlsoap.org/wsdl/	WSDL Schema
wsdl11soap11	http://schemas.xmlsoap.org/wsdl/soap/	WSDL SOAP 1.1 Schema
wsdl11soap12	http://schemas.xmlsoap.org/wsdl/soap12/	WSDL SOAP 1.2 Schema

1.5 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [IETF RFC 2119].

Parenthetic remarks about fault subcodes are mentioned throughout the document where a conformance issue may result in a error. The treatment of these subcodes is described in section 2.9 - Faults. These faults only apply to SOAP exchanges.

1.6 Conformance

A conforming service requester or provider MUST meet the requirements specified in section 2 - The WMQ Service Binding. Conforming service requesters and providers MUST support the WMQ IRI scheme [WMQ-IRI] to the extent required by section 2. A conforming service requester or provider MAY implement the requirements in section 3 - WSDL Usage.

2 The WMQ Service Binding

2.1 Overview

This section defines the properties and rules needed to describe a WMQ application as a Service. It covers both SOAP and non-SOAP WMQ messages, and the MQI structures and APIs used to send the messages. Specifically, this section defines:

- How a WMQ service requester connects to a queue manager and selects the appropriate message destination(s).
- How message properties and header fields (for example `wmqservice:priority` and `wmqservice:targetAction`), MUST be handled by a WMQ service requester and provider.
- How the message content, and properties MUST be handled for SOAP messages by a WMQ service requester and provider.
- The message exchange patterns supported by the WMQ service binding.

Note:

The specification defines addressing for both queues and topics. The generic term 'destination' is used to refer to either.

2.2 WMQ Service Definition Name

The namespace associated with this version of the specification is:

`http://www.ibm.com/xmlns/prod/wmq/bindings/1.0`

2.3 WMQ Generic Binding Properties

The following section defines a set of properties which apply whether a service is using a WMQ SOAP binding or a WMQ Native binding. These properties specify: Connection information, (how and where requesters and responders connect to a WMQ infrastructure); Addressing information, (the destinations to use in a message exchange); Message Exchange Pattern information and Message Correlation styles; and Qualities of Service (how to treat messages in terms of priority, persistence etc.). The properties are grouped according to their function, and their placement in the encoded message.

Property names are always case-sensitive.

2.3.1 Connection to a Destination

This set of properties provides the information necessary to establish a connection to a queue manager and resolve the address of the target service.

[**wmqservice:destinationName**] (string)

Specifies the name of the destination (queue or topic). This name can be left unqualified or can be qualified with a queue manager name. It corresponds to the ObjectName (and ObjectQMgrName) of the MQOD structure used on an MQOPEN() call.

- MUST take the form of a WMQ destination as defined by `queue-dest` or `topic-`

`dest` part of the `wmq:` IRI scheme. E.g.

Unqualified Queue Destination Example
msg/queue/INS.QUOTE.REQUEST

Qualified Queue Destination Example (with Queue Manager Name)
msg/queue/INS.QUOTE.REQUEST@MOTOR.INS

Topic Destination Example
msg/topic/Stocks/Prices/IBM

[**wmqservice:connectQueueManager**] (string)
The name of the queue manager to which the requesting service SHOULD connect to. This corresponds to the QmgrName parameter used on the MQCONN() or MQCONNX() calls.

- The default value is a blank queue manager name, which in server-bindings mode will cause WMQ to connect to the default queue manager, and in client-bindings mode will cause a connection to the queue manager that is listening on the socket.

2.3.2 Client-Binding Connection Properties

This set of properties provides the information necessary to connect to a queue manager in client-binding connection mode. The first two properties: `wmqservice:channelTableName` and `wmqservice:channelTableLib` identify the name and location of the WMQ channel table which may be used to identify the channel connection. They provide the equivalent of the MQCHLTAB and MQCHLLIB environment variables described in the WMQ "Clients" documentation.

If no value is specified for `wmqservice:channelTableName` then the properties `wmqservice:channelName` , `wmqservice:connectionName`, and `wmqservice:transportType` MAY be used to specify the details of a channel connection. They provide the equivalent of the MQSERVER environment variable described in the WMQ Intercommunications manual. To provide a complete channel definition all three of these properties MUST be specified together.

If the WMQ environment variables MQSERVER, MQCHLTAB, or MQCHLLIB have been specified then they MUST take priority over the equivalent `wmqservice` properties.

Note:

This specification allows the creation of WMQ service definitions which describe how consumers can bind to that service. Detailed bindings can include information about how a service requester binds to a specific machine or channel. Being able to specify client-bindings, channels names etc. will be useful in *some* circumstances, but over-specifying the service binding may prove to be restrictive. Service providers are advised to minimise the amount of binding information incorporated into a service definition and to allow to underlying infrastructure or WMQ to route messages where possible.

[**wmqservice:channelTableName**] (string)

This is the name of the client channel table file.

- If this property is specified then the `wmqservice:channelName` , `wmqservice:transportType`, and `wmqservice:connectionName` properties MUST be ignored;
- If either the MQSERVER or MQCHLTAB environment variables are set then this property MUST be ignored.

[**wmqservice:channelTableLib**] (string)

This is the Path to the client channel table

- If either the MQSERVER or MQCHLLIB environment variables are set then this property MUST be ignored;
- If the `wmqservice:channelTableName` property is not specified then this value MUST be ignored.

[**wmqservice:channelName**] (string)

Specifies the channel to be used when a when a WMQ service requester makes a WMQ client-binding connection. This property corresponds to the ChannelName field in the MQCD structure on an MQCONNX call.

- A value MUST be given for this property if the `wmqservice:connectionName` property is specified;
- If `wmqservice:channelTableName` is specified this property MUST be ignored.
- If either the MQSERVER or MQCHLTAB environment variables are set then this property MUST be ignored;
- By default this property is not set.

[**wmqservice:connectionName**] (string)

Specifies the connection string to be used when a service requester makes a WMQ client-binding connection. For TCP/IP, this in the form of a host name which can be followed by a port number separated by a colon. If the port is not specified it will be defaulted by WMQ (to 1414). This property corresponds to the ConnectionName and ShortConnectionName fields in the MQCD structure on an MQCONNX call.

- If `wmqservice:channelTableName` is specified this property MUST be ignored;
- By default this property is not set;
- If either the MQSERVER or MQCHLTAB environment variables are set then this property MUST be ignored.

[**wmqservice:transportType**] (string)

Specifies the transport type to be used when a WMQ service requester makes a WMQ client-binding connection. This property corresponds to the Transport Type in the MQCD structure on an MQCONNX call.

- A value MUST be given for this property if the `wmqservice:connectionName` property is specified;
- If `wmqservice:channelTableName` is specified this property MUST be ignored;
- If either the MQSERVER or MQCHLTAB environment variables are set then this property MUST be ignored;
- This property defaults to MQXPT_TCP;
- Valid values are:
  - `MQXPT_LU62`
    Used to indicate the LU 6.2 transport protocol.
  - `MQXPT_TCP`
    Used to indicate the TCP/IP transport protocol.

2.3.3 WMQ Message Descriptor (MQMD) Header properties

This set of properties provides information that corresponds to the Message Descriptor (MQMD) Header fields.

[**wmqservice:reportOptions**] (string)

Specifies how the message and correlation identifiers in the reply or fault messages are set by the service provider. This property corresponds to the Report field in the MQMD structure.

The value can either be specified as:

- The decimal representation of an integer corresponding to any of the valid MQRO_* values defined for the `MQMD.Report` field in the Application Programming Reference, and WMQ Constants documentation.
  ...or...
- One or more of the text constants listed below (which is a subset of all of the MQRO_* options). If multiple text values are specified then they MUST be separated by commas.

A WMQ service requester can add other options to this value when it sets the MQMD Report field (see sections 2.8.2 and 2.8.3) provided they do not affect the Message-identifier and Correlation-identifier options listed below.

- The recognised values (and text constants) are:
  - Message-identifier options: Specify one of the options listed below to control how the msgId of the report message (or reply message) is to be set.
    - `0 (MQRO_NEW_MSG_ID)`
      This is the default action.
      It indicates that if a report or reply is generated as a result of this

message, a new msgId is generated for the report or reply message.

    - `128 (MQRO_PASS_MSG_ID)`
      If a report or reply is generated as a result of this message, the msgId of this message is copied to the msgId of the report or reply message.

  - Correlation-identifier options: Specify one of the options listed below to control how the correlId of the report message (or reply message) is to be set.

    - `0 (MQRO_COPY_MSG_ID_TO_CORREL_ID)`
      This is the default action. It indicates that if a report or reply is generated as a result of this message, the msgId of this message is copied to the correlId of the report or reply message.

    - `64 (MQRO_PASS_CORREL_ID)`
      If a report or reply is generated as a result of this message, the correlId of this message is copied to the correlId of the report or reply message.

- The default value is 0 (corresponding to `"MQRO_NEW_MSG_ID,MQRO_COPY_MSG_ID_TO_CORREL_ID"`)

[**wmqservice:msgType**] (string)
Indicates the type of the message that will be sent. This property corresponds to the MsgType in the MQMD structure.

- This value can either be specified as:
  - The decimal representation of an integer corresponding to any of the valid MQMT_* values defined for the `MQMD.MsgType` field in the Application Programming Reference, and WMQ Constants documentation.
    ...or...
  - A text constant corresponding to the subset of MQMT_* values described below.

- The recognised values (and text constants) are:
  - `1 (MQMT_REQUEST)`
    The message is one that requires a reply. This value indicates that the service is using a request-response message exchange pattern.

  - `2 (MQMT_REPLY)`
    The message is a reply to a request.

  - `4 (MQMT_REPORT)`
    The message is a report.

  - `8 (MQMT_DATAGRAM)`
    This value indicates that the service is a one-way message exchange. There will not be a reply.

- If this property is not specified, the value SHOULD be set according to the message exchange pattern. See section 2.8.1.1 for information on how to determine the message exchange pattern.

[**wmqservice:expiry**] (int)

This is the message lifetime, specified as the decimal representation of a signed integer, and measured in tenths of seconds.

- Valid values are any number in the range 1 to 2147483647;
- The special value -1 may be used to indicate that message does not expire;
- The default value is -1.

[**wmqservice:format**] (string)

The format name of message data. This property corresponds to the MQRFH2 Format field or (if no MQRFH2 is present) the MQMD format field

- A character string - up to 8 characters long - used to describe the format of the message data.
- For non-SOAP messages this can be set to any value according to the guidelines in the Application Programming Reference. It is recommended that only the characters A-Z and 0-9 are used.
- See section 2.5 - WMQ SOAP Binding Properties for the values which MUST be used in this property for SOAP message payloads.
- If the value of this property is less than 8 characters then the service requester MUST pad it with blanks
- If the value of this property is greater than 8 characters then the service requester MUST fail in a manner appropriate to the requesting environment. [If a SOAP fault is generated as a result of this failure, then fault subcode `wmqservice:malformedFormat` MUST be used];
- If this property is not specified, the WMQ default (a blank value) SHOULD be assumed for Native bindings.

[**wmqservice:priority**] (int)

This indicates the priority associated with the message. It is specified as the decimal representation of an integer. It corresponds to the Priority field in the MQMD structure.

- Valid values are integers between 0 (lowest priority) and 9 (highest priority).
- A special value (-1) can be used to indicate that the message priority is taken from the definition of the first queue the message is put to.
- The default value is -1 (which equates to the constant MQPRI_PRIORITY_AS_Q_DEF).

[**wmqservice:persistence**] (string)

This indicates whether the message is persistent or not. It corresponds to the Persistence field in the MQMD structure.

- This value can either be specified as:
    - The decimal representation of an integer corresponding to the valid MQPER_* values defined for the `MQMD.Persistence` field in the Application Programming Reference, and WMQ Constants documentation.

    - A text constant corresponding to the MQPER_* values described below.

  The recognised values (and text constants) are:

    - `0 (MQPER_NOT_PERSISTENT)`
      Messages will not be persistent.

    - `1 (MQPER_PERSISTENT)`
      Messages will be persistent.

    - `2 (MQPER_PERSISTENCE_AS_Q_DEF)`
      This is the default value. The queue manager will determine the message persistence from the definition of the destination the message is put to.

[**wmqservice:msgId**] (string)
The Message identifier. This property corresponds to the `MsgId` field in the MQMD structure;

- Can either be specified as a character string, or as a binary value:
    - A binary value MUST be prefixed with the characters `0x:` followed by a string of (up to 24 pairs of) two-character hexadecimal values – each of which represents one byte of the msgId;
    - The `0x:` prefix is used only to identify the value as a binary value and MUST NOT be treated as part of the msgId value;
    - If there is an odd number of characters following the `0x:` prefix, or if the following string contains a non-hexadecimal character (i.e. one that is not in the ranges 0-9, a-f, and A-F) then the service requester MUST fail in a manner appropriate to the requesting environment;
    - If the binary value is less than 24 bytes long then a sequence of null (binary zero) bytes MUST be appended to the value to make it 24 bytes long;
    - A character value is considered to be any value which is not prefixed with `0x:`
    - The characters used in a character value MUST be restricted to UTF-8 characters the range: 0x20–0x7E (these are sometime known as the "ASCII printable characters").
    - If a character value is less than 24 bytes long then space characters (code point 0x20) MUST be appended to the value to make it 24 bytes long;
    - If the value of this property (whether character or binary) contains invalid characters or is longer than 24 bytes then the service requester MUST fail in a manner appropriate to the requesting environment.
    - [If a SOAP fault is generated as a result of a badly formatted msgId then fault subcode wmqservice:malformedMsgId MUST be used].

- If this property is not specified the default value of binary zero for the length of the field (MQMI_NONE) is be used to force the `msgId` to be generated by the queue manager.

Note:

The msgId property is included in this specification to allow certain specialised WMQ applications to be described as services (for example applications which share an input queue and select the messages intended for them based on a pre-defined msgId value). Predefined msgId's in service definitions MAY lead to problems (for example when when using request–response MEP's which return the request's msgId in the response's correlId). Predefining the msgId in a service definition is therefore NOT RECOMMENDED outside specialised scenarios.

[**wmqservice:correlId**] (string)

The Correlation identifier . This property corresponds to the `CorrelId` field in the MQMD structure.

- Can either be specified as a character string, or as a binary value:
  - A binary value MUST be prefixed with the characters `0x:` followed by a string of (up to 24 pairs of) two-character hexadecimal values – each of which represents one byte of the msgId;
  - The `0x:` prefix is used only to identify the value as a binary value and MUST NOT be treated as part of the msgId value;
  - If there is an odd number of characters following the `0x:` prefix, or if the following string contains a non-hexadecimal character (i.e. one that is not in the ranges 0-9, a-f, and A-F) then the service requester MUST fail in a manner appropriate to the requesting environment;
  - If the binary value is less than 24 bytes long then a sequence of null (binary zero) bytes MUST be appended to the value to make it 24 bytes long;
  - A character value is considered to be any value which is not prefixed with `0x:`
  - The characters used in a character value MUST be restricted to UTF-8 characters the range: 0x20–0x7E .
  - If a character value is less than 24 bytes long then space characters (code point 0x20) MUST be appended to the value to make it 24 bytes long;
  - If the value of this property (whether character or binary) contains invalid characters or is longer than 24 bytes then the service requester MUST fail in a manner appropriate to the requesting environment.
  - [If a SOAP fault is generated as a result of a badly formatted correlId then fault subcode wmqservice:malformedCorrelId MUST be used].

- The default value of this property is binary zero for the length of the field (MQCI_NONE).

[**wmqservice:replyTo**] (string)

Specifies the name of the destination (queue or topic) to which the response message MUST be sent. It corresponds to the ReplyToQ (and ReplyToQMgr) of the MQMD structure.

- MUST take the form of either a full WMQ IRI address including the IRI scheme name (`wmq`), or the `queue-dest` or `topic-dest` particle of a WMQ IRI, as defined by the `wmq:` IRI scheme.
    - *Example of full WMQ IRI address*
      `wmq://branch452.example.com/msg/queue/INS.QUOTE.REPLY?persistence=MQPER_NOT_PERSISTENT&priority=9`
    - *Example of partial address*
      `msg/queue/INS.QUOTE.REPLY`
- If a topic IRI is used it MUST be no more than than 48 characters.

[**wmqservice:codedCharSetId**] (int)
Character encoding of message data. It is specified as the decimal representation of an integer. This property corresponds to the `CodedCharSetId` field in the MQMD structure.

- If this property is not specified the service requester and service provider MUST use a value which corresponds to the character set of the message data.

Note:

The MQMD `encoding` field is defined as a property in section 2.4 - WMQ Native Binding Properties.

The following MQMD fields are not defined as properties by this specification: `StrucId, Version, Feedback, BackoutCount, UserIdentifier, AccountingToken, ApplIdentityData, PutApplType, PutApplName, PutDate, PutTime, ApplOriginData, GroupId, MsgSeqNumber, Offset, MsgFlags, OriginalLength`

2.3.4 MQRFH2 Message properties

An MQRFH2 header MUST be used in some circumstances in order to convey all the information required by a WMQ service message. The `wmqservice` MQRFH2 MUST be included if ANY of the following conditions are true:

- The `wmqservice:replyTo` IRI contains more information than can be stored in the request message's MQMD ReplyToQ and ReplyToQMgr fields (for example it contains properties);
- The `wmqservice:targetAction` property is specified (this is a WMQ Native Binding property - see section 2.4);
- The `wmqservice:soapAction` or `wmqservice:contentType` properties are specified (these are WMQ SOAP Binding properties- see section 2.5);
- User-defined (unrecognised) properties are specified (see section 2.3.5).

If a `wmqservice` MQRFH2 header is needed for SOAP messages then it MUST be the last WMQ header structure before the payload.

The following MQRFH2 property applies to both WMQ Native and SOAP bindings.

[**wmqservice:replyTo**] (string)

This is the same property as that described in section 2.3.3 it specifies the name of the target destination (queue or topic) for response messages. It is applied to the RFH2 as follows:

- If this property is to be included in the `wmqservice` MQRFH2 then it MUST appear in the <mq_svc> folder as a property named `replyTo`.
- This property SHOULD be set to the full value of the `wmqservice:replyTo` IRI.

2.3.5 User Defined Properties

A WMQ service definition may include one or more user-defined properties in either the IRI or the WSDL.

[**wmqservice:usr**] (string)

User-defined data carried in WMQ service messages. See section 2.7.3.2 for information on how these properties are encoded in the message.

- Specified in the IRI by prefixing a property name (or property names) with the text `'usr'`
- Specified in the WSDL as a value (or values) in the `wmqservice:usr` element. If specified in WSDL, the value of the `wmqservice:usr` element MUST conform to the the content model for an MQRFH2 folder as documented in the WMQ Application Programming Reference.
- The specification of security sensitive properties (e.g. userId and/or password) in the IRI or WSDL is strongly discouraged.

2.4 WMQ Native Binding Properties

The following section describes the properties which apply only to Native WMQ service bindings.

[**wmqservice:encoding**] (int)

Numeric encoding of message data. This property corresponds to the Encoding field in the MQMD structure.

- This value is specified as the decimal representation of an integer corresponding to the valid MQENC_* values defined for the `MQMD.Encoding` field in the Application Programming Reference documentation;
- If this property is not specified the sender can choose an appropriate encoding, or can allow WMQ to use its default value;
- This property is only allowed for native bindings. If the `wmqservice:encoding` property is specified for for SOAP bindings then the service requester or service provider MUST fail in a manner appropriate to the requesting environment. [If a SOAP fault is generated as a result then fault subcode wmqservice:encodingNotAllowedForSoapBindings MUST be used];

[**wmqservice:targetAction**] (string)

Used by the service provider to dispatch service requests. For example, this property allows multiple services or operations to be deployed using a single destination, and a service provider to dispatch the requests arriving on that destination appropriately.

- If specified MUST appear in the `<mq_svc>` folder as a property named `targetAction;`
- This property is only allowed for native bindings. If `wmqservice:targetAction` is specified for SOAP bindings then the service requester or service provider MUST fail in a manner appropriate to the requesting environment. [If a SOAP fault is generated as a result then fault subcode wmqservice:targetActionNotAllowedForSoapBindings MUST be used];

2.5 WMQ SOAP Binding Properties

The following section describes the properties and considerations which apply only to SOAP WMQ service bindings.

2.5.1 Format Considerations

When transporting a SOAP message there is a requirement to identify the type and version of the SOAP payload without necessarily parsing the message body. The Format field in the appropriate WMQ header structures (i.e. final WMQ header before the SOAP payload) MUST be used for this purpose. This section defines the rules which MUST be applied to `wmqservice:format` to determine the SOAP payload type.

For SOAP payloads using the WMQ SOAP Binding, the `wmqservice:format` property MUST be set by the service requester or service provider to one of the following pre-defined values. The list refers to the `wmqservice:contentType` property which is defined in section 2.5.2.

- `MQSOAP11`
  For XML SOAP 1.1 messages.
  This is equivalent to setting the `wmqservice:contentType` property to `text/xml`
- `MQSOAP12`
  For XML SOAP 1.2 messages.
  This is equivalent to setting the `wmqservice:contentType` property to `application/soap+xml`
- `MQSWA11`
  For SOAP with Attachments messages.
  This is equivalent to setting the `wmqservice:contentType` property to `multipart/related; type="text/xml"`
- `MQSWA12`
  For SOAP with Attachments messages.
  This is equivalent to setting the `wmqservice:contentType` property to `multipart/related; type="application/soap+xml"`
- `MQMTOM11`

For MTOM messages
This is equivalent to setting the `wmqservice:contentType` property to `multipart/related; type="application/xop+xml"; start-info="text/xml"`

- `MQMTOM12`
  For MTOM messages.
  This is equivalent to setting the `wmqservice:contentType` property to `multipart/related; type="application/xop+xml" start-info="application/soap+xml"`

2.5.2 MQRFH2 Message Properties for SOAP Binding

As described in 2.3.4 - MQRFH2 Message properties, a `wmqservice` MQRFH2 header might be needed to convey all the properties of the SOAP/WMQ request or response. The following MQRFH2 properties apply specifically to WMQ messages with SOAP payloads:

[**wmqservice:soapAction**] (string)
This property corresponds to SOAPAction in SOAP/HTTP bindings (see [SOAP 1.1]).

- If the `wmqservice` MQRFH2 header is included for a SOAP message and this property is specified, then it MUST appear in the `<mq_svc>` folder as a property named `soapAction` .

- If using SOAP 1.2, and the `wmqservice:contentType` property has an `action` parameter, that parameter value MUST match this `mq_svc.soapAction` value. [Use fault subcode **mismatchedSoapAction** if the SOAP 1.2 `action` does not match]

- This property is only allowed for SOAP bindings. If `wmqservice:soapAction` is specified for native bindings then the service requester or service provider MUST fail in a manner appropriate to the requesting environment.

[**wmqservice:contentType**] (string)
The `contentType` value allows specification of additional information about the MIME type of the primary message payload. It has the same values as the MIME Content-Type specified for a SOAP message over HTTP [RFC 2045].

- If the `wmqservice` MQRFH2 header is included for a SOAP message then this property MAY be included in the MQRFH2. If this property is included then it MUST appear in the `<mq_svc>` folder as a property named `contentType` .

- If specified, this value MUST contain the MIME equivalent to the `wmqservice:format` value in the `wmqservice` MQRFH2 header `Format` field described above. For example, if the Format field is set to, `MQSOAP11,` the `wmqservice:contentType` property MUST be set to "text/xml". If the `wmqservice` MQRFH2 header `Format` field and the `wmqservice:contentType` property are inconsistent, the requester or provider MUST fail in a manner appropriate to the environment and context. [If a SOAP fault is generated as a result of this failure, then fault subcode `wmqservice:mismatchedContentType` MUST be used].

- If the `charset` parameter is specified for `wmqservice:contentType` then it SHOULD match the `encoding` attribute of the SOAP envelope's XML declaration (*"<?xml"*), and SHOULD be equivalent to the CodedCharSetId field in the final WMQ header (MQMD or

MQRFH2). See 2.7 - WMQ Message Construction for more information about the precedence rules associated with character sets and CCSIDs.

- For SwA payloads the `type` parameter MUST be set according to the values in section 2.5.1 - Format Considerations (see section 2.7.2.2 SOAP Messages with Attachments for more information).

- For MTOM payloads, the `type` and `start-info` parameters MUST be set according to the values in section 2.5.1 (see section 2.7.2.2 SOAP Messages with Attachments for more information).

2.6 Binding of Properties to IRI

WMQ service requesters or service providers need to allow for the setting of the above properties. Some properties, as mentioned above can be inferred from context, or provided by the application environment. Some can be put into WSDL.

In many cases, it is desirable to represent those properties as part of the URL-like representation described in [WMQ-IRI]. The following table describes how the properties above are used in the IRI. For brevity, `wmqservice` properties are shown without the `wmqservice` prefix.

wmqservice Property	IRI Representation
destinationName	as `queue-dest` or `topic-dest` part of the IRI syntax
connectQueueManager	as `connectQueueManager` query parameter
channelTableName	as `channelTableName` query parameter
channelTableLib	as `channelTableLib` query parameter
channelName	as `channelName` query parameter
connectionName	as `connectionName` portion of IRI syntax
transportType	as `transportType` query parameter
reportOptions	as `reportOptions` query parameter
msgType	as `msgType` query parameter
expiry	as `expiry` query parameter
format	as `format` query parameter
priority	as `priority` query parameter
persistence	as `persistence` query parameter
msgId	as `msgId` query parameter
correlId	as `correlId` query parameter
replyTo	as `replyTo` query parameter
codedCharSetId	as `codedCharSetId` query parameter
encoding	as `encoding` query parameter
targetAction	as `targetAction` query parameter
soapAction	as `soapAction` query parameter
contentType	Not allowed in IRI
usr	as property name prefixed with '`usr`'

2.7 WMQ Message Construction

WebSphere MQ messages conforming to this specification can use the Native WMQ Binding, or the SOAP WMQ Binding as described below.

2.7.1 Native WMQ Binding Messages

For the Native WMQ Binding, messages are composed of an MQMD structure; a chain of WMQ header structures if needed (typically MQRFH or MQRFH2); and finally the actual payload of the message. The WMQ header structures MAY include the `wmqservice` MQRFH2 header (described in see section 2.3.4) and MAY also include other WMQ headers – for example an MQRFH2 header carrying publish/subscribe information.

Normal WMQ rules MUST be applied to govern the chaining of headers and the conversion of character set and numeric encoding data.

In some cases WMQ headers can be considered to be part of the request payload itself. For instance if the service definition describes a CICS or IMS application then the `wmqservice:format` MAY be set to `MQCICS` or `MQIMS`, in which case the headers can be considered to be part of the message payload.

Note:

The WMQ service definition allows existing WMQ applications to be described as services. These applications will typically not be aware of the WMQ service definition, and so will not expect messages to be prefixed with a service definition MQRFH2. When creating service definitions for these applications, take care to avoid the properties described in sections 2.3.4 and 2.5.2 which will cause the message to be prefixed with an MQRFH2 header.

2.7.2 SOAP WMQ Binding Messages

For the SOAP WMQ Binding, messages are composed of an MQMD structure; a chain of WMQ header structures if needed (typically MQRFH or MQRFH2); and finally a SOAP payload which is formatted as described in this section.

The WMQ header structures MAY include the `wmqservice` MQRFH2 header (described in see section 2.3.4) and MAY also include other WMQ headers – for example an MQRFH2 header carrying publish/subscribe information. Normal WMQ rules MUST be applied to govern the chaining of headers and the conversion of character set and numeric encoding data.

The encoding of the SOAP payload in the message body depends on the SOAP version, and whether there are any attachments. The `wmqservice:format` property of the WMQ header structure immediately preceding the payload (and `wmqservice` MQRFH2 `wmqservice:contentType` property if specified) indicate the encoding of the SOAP payload as described in the preceding section (2.5 - WMQ SOAP Binding Properties and the following subsections).

2.7.2.1 SOAP without Attachments

For SOAP messages without attachments the payload MUST consist of the properly encoded bytes of the XML SOAP message, and the `Format` field in the last WMQ header MUST be "MQSOAP11" (for SOAP 1.1) or "MQSOAP12" (for SOAP 1.2). If there is a `wmqservice:contentType` property in a `wmqservice` MQRFH2 header then its content type value MUST be "text/xml" (for SOAP 1.1) or "application/soap+xml" (for SOAP 1.2) (other parameter values such as `charset` are allowed in the `wmqservice:contentType)`.

2.7.2.2 SOAP Messages with Attachments

For SOAP messages with attachments the payload MUST consist of a multipart MIME message and the `Format` field in the last WMQ header MUST be set to one of the following formats from section 2.5.1 - Format Considerations.

Format	Message Encoding
MQSWA11	SOAP Messages with Attachments [SwA] (SOAP 1.1 Payload)
MQSWA12	SOAP Messages with Attachments [SwA] (SOAP 1.2 Payload)
MQMTOM11	XOP [SOAP11-MTOM]
MQMTOM12	XOP [SOAP12-MTOM]

If the `wmqservice:contentType` property is specified the content-type itself MUST be '`multipart/related`'. If `wmqservice:contentType` property is not specified and the Format of the message is one of the Formats listed above, then '`multipart/related`' MUST be inferred.

Multipart MIME messages normally have several parameters that describe the structure of the data being sent. These parameters MAY be specified explicitly in a `wmqservice:contentType` property in the `wmqservice` MQRFH2 header or, if they are not specified explicitly, they MUST be inferred from the message itself as described below:

- `start`
  If this parameter is explicitly specified then it MUST indicate the 'Body Part' of the MIME message. If it is not specified then the start parameter MUST be inferred from the message by assuming that the first MIME part encountered in the message body MUST be what section 2.5 of the MIME Part One specification [RFC 2045] calls a "Body Part", and this part MUST contain the SOAP payload.
- `boundary`
  If this parameter is explicitly specified then it MUST indicate the boundary parameter from the MIME Part Two specification [RFC 2046]. If it is not specified then it MUST be inferred from the message by assuming that the first line encountered in the message body MUST be parsed as the MIME boundary.

The value of the following parameters vary depending on the `wmqservice:format`.

If the `wmqservice:format` is MQSWA11 or MQSWA12 then the following parameter applies:

- `type`
  If this property is explicitly specified then it MUST match the values in section 2.5.1 - Format Considerations. If it is not specified then it MUST be inferred from the `wmqservice:format` property according to the rules in section 2.5.1.

If the `wmqservice:format` is MQMTOM11 or MQMTOM12 then the following parameters apply:

- `type`
  If this parameter is explicitly specified then it MUST be set to 'application/xop+xml'. If it is

not specified explicitly then ‘application/xop+xml’ MUST be inferred

- `start-info`
  If this parameter is explicitly specified it MUST match the values in section 2.5.1 - Format Considerations for the particular `wmqservice:format` being used. If it is not specified explicitly then ‘text/xml’ or ‘application/soap+xml’ MUST be inferred from the `wmqservice:format` property according to the rules in section 2.5.1.

2.7.2.3 Character Conversion for SOAP messages

The following precedence rules MUST be applied to determine the character set used when interpreting SOAP messages:

- The character set MUST be determined by the `charset` parameter in the `wmqservice` MQRFH2 header `mq_svc.contentType` property if it is present.
- If there is no `charset` parameter in the `wmqservice` MQRFH2 header `mq_svc.contentType` property, then the service requester or provider SHOULD attempt to process the SOAP message using the `CodedCharSetId` field in the WMQ header structure preceding the SOAP payload. If the document does not appear to be correctly encoded using the character set described by the appropriate WMQ `CodedCharSetId` field, then a message recipient MAY attempt to infer the character set of XML SOAP payloads using the rules defined at Autodetection of Character Encodings from [XML 1.0].
- If the character set of the SOAP message still cannot be determined then the recipient of the SOAP request MUST fail in a manner appropriate to the environment and context [If a SOAP fault is generated as a result then fault subcode wmqservice:unknownCharacterSet MUST be used].

2.7.3 MQRFH2 Header Construction

There are circumstances (for both SOAP and Native WMQ service bindings) when a MQRFH2 header is REQUIRED to carry `wmqservice` property information (see section 2.3.4 - MQRFH2 Message properties).

When a `wmqservice` MQRFH2 header is needed, properties in the header MUST be stored as property triplets (see the WMQ "Using Java" documentation) in a root folder named `<mq_svc>`. The data type specifier portion of the triplet is optional for the MQRFH2 properties described in sections 2.3.4 and 2.5.2 because they are “well known” properties with pre-defined types.

The `wmqservice` MQRFH2 is also used to carry a binding version and user defined properties:

2.7.3.1 Binding Version

The `<mq_svc>` folder in the `wmqservice` MQRFH2 header MUST contain the following property to identify the version of the WMQ service definition specification.

[**wmqservice:bindingVersion**] (string)
This documents the WMQ Service Definition binding version.

- For version 1.0 (http://www.ibm.com/xmlns/prod/wmq/bindings/1.0) of this specification, the `bindingVersion` MQRFH2 property MUST be set to the value `"1.0"`.

```
e.g.  <bindingVersion>1.0</bindingVersion>
```

- If a service requester or service provider receives a message with an unrecognised binding version then the requester or provider MUST fail in a manner appropriate to the environment and context. [If a SOAP fault is generated as a result of this failure, then fault subcode `wmqservice:unrecognizedBindingVersion` MUST be used].

2.7.3.2 User Defined Properties

User defined properties can be specified in a WMQ service definition IRI, or in WSDL. These are carried in the `wmqservice` MQRFH2 as follows:

- If there are query parameter properties in the IRI which are prefixed with the text `'usr'`, then the service requester SHOULD include these as properties in a subfolder named `<usr>` in the `wmqservice` MQRFH2 `<mq_svc>` folder. If a user-defined property is included in the message then the `'usr'` prefix MUST be removed from the name of the property.
- The value (or values) of any `wmqservice:usr` properties specified in the WSDL, SHOULD be included in a subfolder named `<usr>` in the `wmqservice` MQRFH2 `<mq_svc>` folder.

For example, the IRI:

```
wmq:/msg/queue/INS.QUOTE.REQUEST?usrProp1=xyz&usrProp2=abc
```

...SHOULD result in the MQRFH2 folders:

```
<mq_svc>
  <bindingVersion>1.0</bindingVersion>
  <usr>
    <Prop1>xyz</Prop1>
    <Prop2>abc</Prop2>
  </usr>
</mq_svc>
```

Service requesters and service providers MAY also define their own properties and include them in the `<usr>` folder.

2.8 Message Exchange Patterns

2.8.1 Overview

A Message Exchange Pattern (MEP) describes the number and direction of messages sent and received during the invocation of a service. WMQ service requesters and service providers conforming to this specification MUST support the Request-Response and One-way message exchange patterns specified in sections 2.8.2 and 2.8.3.

[SOAP 1.1] does not have a formal specification of Message Exchange Patterns, so when sending SOAP 1.1 messages a conforming WMQ service requester or service provider MUST support the "Request-Response" and "One-Way" patterns specified in this document.

In the case of SOAP 1.2 a conforming WMQ service requester or provider MUST support the following message exchange patterns:

- `http://www.w3.org/2003/05/soap/mep/request-response/` (see 6.2 Request-Response Message Exchange Pattern from [SOAP 1.2 Part 2])

- `http://www.w3.org/2006/08/soap/mep/one-way/` (see [SOAP 1.2 Part 3])

Where this section discusses specific SOAP roles and properties, equivalent roles and properties are assumed for a native (non-SOAP) WMQ bindings.

2.8.1.1 Determining the MEP

When a WSDL document is used to describe a WMQ service then the MEP can be determined from the WSDL as described in [WSDL 1.1]. When an IRI is taken from some context like the replyTo property, or WS-Addressing, then the MEP can be determined from that context. However, when a service is described using only an IRI, and the MEP cannot be determined from the IRI context, then the MEP MUST be determined using the following rules:

1. If the wmqservice:msgType property is present and its value is MQMT_REQUEST then a request-response message exchange pattern MUST be used
2. If the wmqservice:msgType property is present and set to any other value, then a one-way message exchange pattern MUST be used
3. If the wmqservice:msgType property is not present and the wmqservice:replyTo property is present then a request-response message exchange pattern MUST be used
4. If neither the wmqservice:msgType property nor the wmqservice:replyTo property are present, then a one-way message exchange pattern MUST be used

2.8.1.2 Using WMQ Service Properties in Message Exchanges

The message construction tables in the following sections 2.8.2 and 2.8.3 show the WMQ message header fields and properties which MUST be set in a message exchange. Only the fields and properties affected by this specification have been included - other fields and properties will continue to follow normal MQI behaviour (for instance, the `MQMD.PutDate` value will be set by the queue manager).

Properties can be obtained from a number of sources, including:

1. A WMQ IRI (which may be specified programmatically, retrieved from the environment etc.);
2. WSDL elements or attributes (including the IRI which can be set in the WSDL address element's 'location' attribute) (see section 3).

It is also possible for a service requester or service provider to override properties if necessary.

The following table shows which properties can be set in the IRI and WSDL, and whether the property is allowed, disallowed, or mandatory in that source. For brevity, `wmqservice` properties are shown without the `wmqservice` prefix.

`wmqservice` Property	Allowed in IRI?	Allowed in WSDL?
destinationName	Mandatory	Mandatory (in IRI)
connectQueueManager	Yes	Yes
channelTableName	Yes	Yes
channelTableLib	Yes	Yes
channelName	Yes	Yes
connectionName	Yes	Yes
transportType	Yes	Yes
reportOptions	Yes	Yes

msgType	Yes	Yes
expiry	Yes	Yes
format	Yes	Yes
priority	Yes	Yes
persistence	Yes	Yes
msgId	Yes	Yes
correlId	Yes	Yes
replyTo	Yes	Yes
codedCharSetId	Yes	Yes
encoding	Yes	Yes
targetAction	Yes	Yes
soapAction	Yes	Yes
contentType	No	No
usr	Yes	Yes

2.8.2 Request-Response MEP

The following section describes the request-response MEP in terms of the state transitions of the requesting and responding nodes. The description is modelled on the [SOAP 1.2 Part 2] state machine description (section 6.2), however the description in this specification applies to the Native WMQ bindings, and both SOAP 1.1 and SOAP 1.2 (unless stated otherwise).

For service requesters and providers conforming to this specification a node instantiated at the sending interface takes the role (i.e. the property `context:Role`) of `RequestingSOAPNode` and a node instantiated at the receiving and sending interface takes the role (i.e. the property `context:Role`) of `RespondingSOAPNode`.

The remainder of this section consists of descriptions of the MEP state machine. In the following state descriptions, the states are defined as values for the `context:State` property.

Failure reasons as specified in the tables represent values of the property `context:FailureReason` and their values are QNames. If a requesting or responding node enters the `Fail` state, the `context:FailureReason` property will contain the value specified for the particular transition.

2.8.2.1 Behaviour of Requesting Node

Init

In the `Init` state, a WMQ request message is formulated. The following table specifies how the properties described in this specification MUST be used in the message.

Field	Value Set by Conforming Service Requester [*‡]
WMQ MQMD Structure	
Report	Derived from the `wmqservice:reportOptions` property. If the `wmqservice:reportOptions` were specified as text constants then the WMQ service requester MUST convert them into the corresponding integer values, and combine them using a bitwise OR operation.
MsgType	For the request-response message exchange this MUST be set to `MQMT_REQUEST`
Expiry	MUST be set to the value of the `wmqservice:expiry` property
Encoding[†]	MUST be set to the value of the `wmqservice:encoding` property and any binary-encoded numeric data in the payload MUST be encoded to this value. If the `wmqservice:encoding` property is not set then the requesting node MUST set this value to match the numeric encoding of the payload.
CodedCharSetId[†]	MUST be set to the value of the `wmqservice:codedCharSetId` property, and character data in the payload MUST be encoded to this value. If the `wmqservice:codedCharSetId` property is not set then the requesting node MUST set this value to match the character set encoding of the payload.
Format[†]	For SOAP bindings this MUST be one of the Format values defined in 2.5.1 Format Considerations; For non-SOAP bindings it MUST be set to the value of the `wmqservice:format` property.
Priority	MUST be set to the value of the `wmqservice:priority` property
Persistence	MUST be set to the value of the `wmqservice:persistence` property
MsgId	MUST be set to the value of the `wmqservice:msgId` property
CorrelId	MUST be set to the value of the `wmqservice:correlId` property
ReplyToQ	Derived from the `wmq-resource` part of the `wmqservice:replyTo` property. If `wmqservice:replyTo` is not specified, then the requesting node MAY supply a (temporary or otherwise) ReplyToQ for this request message. If the `wmqservice:replyTo` IRI contains information which cannot be carried in the MQMD (for example it contains properties) then this field MUST contain only the `wmq-resource` destination name, and the `wmqservice:replyTo` property in a `wmqservice` MQRFH2 header MUST carry the full reply information.
ReplyToQMgr	Derived from the `wmq-qmgr` part of the `wmqservice:replyTo` property. If the requesting node supplied a (temporary or otherwise) ReplyToQ (as described above), then it MAY also assign a value for this field to identify the ReplyToQ's queue manager.
MQRFH2 Message Header Fields and Properties (if required) [*‡]	
<mq_svc> targetAction	MUST be set to the value of the `wmqservice:targetAction` property if specified
<mq_svc> bindingVersion	MUST be set to "1.0" for specification level http://www.ibm.com/xmlns/prod/wmq/bindings/1.0
<mq_svc><usr>	Derived from the IRI according to the rules in section 2.7.3.2 - User Defined Properties

Field	Value Set by Conforming Service Requester [*‡]
<mq_svc> contentType	MUST be set to the value of the `wmqservice:contentType` property as defined in section 2.5.2.
<mq_svc> soapAction	MUST be set to the value of the `wmqservice:soapAction` property if specified.
<mq_svc> replyTo	MUST be set to the value of the `wmqservice:replyTo` property IRI if the `wmqservice:replyTo` property contains information which cannot be carried in the MQMD (see section 2.3.3 on page 14).
Message Body	
Body	The message from `mep:outboundMessage` serialised according to the `wmqservice:format` property (and media type specified in the `wmqservice` MQRFH2 `wmqservice:contentType` property if required).

Notes:
[*] The values of MQMD and MQRFH2 header fields not specified in the table take their defaults

[‡] If a `wmqservice` property is referenced in this table but not specified in a service definition then use the default value specified in section 2.3, 2.4, or 2.5.

[†] If this WMQ header is followed by another WMQ header (or the `wmqservice` MQRFH2), then normal WMQ header chaining rules apply. The value in this field applies to WMQ header which follows immediately after this one, and the corresponding field in the final WMQ header before the payload MUST be set to the value shown in this table.

An unconditional transition is made from `Init` state to `Requesting` state.

Requesting

In the `Requesting` state, the requester MUST send the message to the destination specified in the `wmqservice:destinationName` property, and wait (either synchronously or asynchronously) for the correlated response message to arrive on the destination specified in the `wmqservice:replyTo` property.

If, the message can't be sent (for example an invalid destination or channel parameters), then a failure reason `fail:TransmissionFailure` is set and a transition to `Fail` is made.

The rules for identifying the correlated response message vary depending on the value of the MQMD `Report` field. See the description of the `wmqservice:reportOptions` property for details.

If a correlated response message is received then the message is stored in `mep:InboundMessage` and a transition to `Sending+Receiving` is made.

If, for some reason (for example a timeout), no correlated response message is received then a failure reason `fail:ReceptionFailure` is set and a transition to `Fail` is made.

Sending+Receiving

In the `Sending+Receiving` state a correlated message has been received which contains either a native WMQ message, or a SOAP envelope.

If a well formed response message is received a transition to `Success` is made.

For SOAP messages, if there is an error in the construction of the message – for example a mismatch between the value of the `wmqservice:contentType` property and the content of the message itself, or a malformed message, then the failure reason MUST be set appropriately and a transition to `Fail` MUST be made.

Success and Fail

`Success` and `Fail` are the terminal states of the Request-Response MEP. Control over the message exchange context returns to the caller.

2.8.2.2 Behaviour of Responding Node

Init

In the `Init` state the inbound request message is received and made available in `mep:InboundMessage`.

An unconditional transition is made from `Init` state to `Receiving` state.

Receiving

If a malformed message has been received then then failure reason MUST be set appropriately and a transition to `Fail` MUST be made.

If the message is well formed then a transition to `Receiving+Sending` MUST be made.

Receiving+Sending

In the `Receiving+Sending` state the response message (reply or fault) is formulated and sent. The following table specifies how the response message is constructed using the properties described in this specification. In the absence of any overriding addressing information (for example WS-Addressing information) the message MUST be sent to the destination specified in the `wmqservice:replyTo` property. If a `wmqservice:replyTo` value is specified in a `wmqservice` MQRFH2 header then it MUST be used in preference to the MQMD ReplyTo and ReplyToQMgr values. The value of the `wmqservice:correlId` and `wmqservice:msgId` properties MUST be set according to the `wmqservice:reportOptions` property of the request message.

Field	Value Set by Conforming Responding Node * ‡
WMQ MQMD Structure	
MsgType	MUST be set to `MQMT_REPLY` if this is the reply to a request message. If this message is not a reply (e.g. it is a fault) then this MAY be set to another value – for example `MQMT_REPORT`.
Expiry	MUST be set to the `wmqservice:expiry` property if it has been specified in the `wmqservice:replyTo` IRI.
Encoding†	MUST be set to the `wmqservice:encoding` property if it has been specified in the `wmqservice:replyTo` IRI, and any binary-encoded numeric data in the payload MUST be encoded to this value. If not specified then the responder MUST set this value to match the numeric encoding of the payload.
CodedCharSetId†	MUST be set to the `wmqservice:codedCharSetId` property if it has been specified in the `wmqservice:replyTo` IRI, and character data in the payload

Field	Value Set by Conforming Responding Node [*] [‡]
	MUST be encoded to this value. If the `wmqservice:codedCharSetId` property is not set then the responder MUST set this value to match the character set encoding of the payload.
Format[†]	For SOAP bindings this MUST be one of the Format values defined in 2.5.1 - Format Considerations. For non-SOAP bindings it MUST be set to a value which is consistent with the response payload.
Priority	MUST be set to the `wmqservice:priority` if it has been specified in the `wmqservice:replyTo` IRI.
Persistence	MUST be set to the `wmqservice:persistence` value if it has been specified in the `wmqservice:replyTo` IRI.
MsgId	MUST be set as determined by the `wmqservice:reportOptions` property in the request message
CorrelId	MUST be set as determined by the `wmqservice:reportOptions` property in the request message
MQRFH2 Message Header Fields and Properties (if required) [*] [‡]	
<mq_svc> bindingVersion	MUST be set to "1.0" for specification level http://www.ibm.com/xmlns/prod/wmq/bindings/1.0
<mq_svc><usr>	MUST be set to the value of the user defined properties if specified in the `wmqservice:replyTo` IRI
<mq_svc> contentType	Determined by the responder according to the rules in section 2.5.2.
Message Body	
body	The message from `mep:outboundMessage` serialised according to the `wmqservice:format` property (and media type specified in the `wmqservice` MQRFH2 `wmqservice:contentType` property if required).

Note:

[*] The values of MQMD and MQRFH2 header fields not specified in the table take their defaults

[‡] If a `wmqservice` property is referenced in this table but not specified in a service definition then use the default value specified in section 2.3, 2.4, or 2.5.

[†] If this WMQ header is followed by another WMQ header (or the `wmqservice` MQRFH2), then normal WMQ header chaining rules apply. The value in this field applies to WMQ header which follows immediately after this one, and the corresponding field in the final WMQ header before the payload MUST be set to the value shown in this table.

If a response message is successfully sent a transition to `Success` is made.

If there is a failure to send a response message then failure reason is set appropriately and a transition to `Fail` is made.

Success and Fail

`Success` and `Fail` are the terminal states for the Request-Response MEP. From the point-of-view of the local node this message exchange has completed.

2.8.3 One-way Message Exchange Pattern

The following section describes the one-way message exchange pattern. This MEP is based on the http://www.w3.org/2006/08/soap/mep/one-way/ message pattern described in[SOAP 1.2 Part 3] .

This MEP applies to Native WMQ bindings, and both SOAP 1.1 and SOAP 1.2 (unless stated otherwise).

This message exchange pattern is identified by the URI `http://www.w3.org/2006/08/soap/mep/one-way/` from [SOAP 1.2 Part 3].

For binding instances conforming to this specification:

- A node instantiated at the sending interface takes the role (i.e. the property `context:Role` from the table of properties describing MEP's) of `SendingSOAPNode` .
- A node instantiated at the receiving interface takes on the role (i.e. the property `context:Role`) of `ReceivingSOAPNode` .

2.8.3.1 Behaviour of Sending Node

The sending node attempts to send the message provided in mep:OutboundMessage to the destination specified in the `wmqservice:destinationName` property. If the Sender fails to send the message then the failure reason is set and a transition to `Fail` is made.

The following table specifies how the properties described in this specification are used in the construction of the message.

Field	Value Set by Conforming Service Requester * ‡
WMQ MQMD Structure	
Report	Derived from the `wmqservice:reportOptions` property. If the `wmqservice:reportOptions` were specified as text constants then the WMQ service requester MUST convert them into the corresponding integer values, and combine them using a bitwise OR operation.
MsgType	MUST be taken from the `wmqservice:msgType` property. If the property is not set this MUST be set to `MQMT_DATAGRAM` but other values associated with one-way messages MAY be used.
Expiry	MUST be set to the value of the `wmqservice:expiry` property
Encoding†	MUST be set to the value of the `wmqservice:encoding` property and any binary-encoded numeric data in the payload MUST be encoded to this value. If the `wmqservice:encoding` property is not set then the requesting node MUST set this value to match the numeric encoding of the payload.
CodedCharSetId†	MUST be set to the value of the `wmqservice:codedCharSetId` property, and character data in the payload MUST be encoded to this value. If the `wmqservice:codedCharSetId` property is not set then the requesting node MUST set this value to match the character set encoding of the payload.
Format†	For SOAP bindings this MUST be one of the Format values defined in 2.5.1 Format Considerations; For non-SOAP bindings it MUST be set to the value of the `wmqservice:format` property.
Priority	MUST be set to the value of the `wmqservice:priority` property
Persistence	MUST be set to the value of the `wmqservice:persistence` property

Field	Value Set by Conforming Service Requester [*‡]
MsgId	MUST be set to the value of the `wmqservice:msgId` property
CorrelId	MUST be set to the value of the `wmqservice:correlId` property
MQRFH2 Message Header Fields and Properties (if required) [*‡]	
<mq_svc> targetAction	MUST be set to the value of the `wmqservice:targetAction` property if specified
<mq_svc> bindingVersion	MUST be set to "1.0" for specification level http://www.ibm.com/xmlns/prod/wmq/bindings/1.0
<mq_svc><usr>	Derived from the IRI according to the rules in section 2.7.3.2 - User Defined Properties
<mq_svc> contentType	MUST be set to the value of the `wmqservice:contentType` property. See section 2.5.2.
<mq_svc> soapAction	MUST be set to the value of the `wmqservice:soapAction` property if specified
Message Body	
body	The message from `mep:outboundMessage` serialised according to the `wmqservice:format` property (and media type specified in the `wmqservice` MQRFH2 `wmqservice:contentType` property if required).

Note:

[*] The values of MQMD and MQRFH2 header fields not specified in the table take their defaults

[‡] If a `wmqservice` property is referenced in this table but not specified in a service definition then use the default value specified in section 2.3, 2.4, or 2.5.

[†] If this WMQ header is followed by another WMQ header (or the `wmqservice` MQRFH2), then normal WMQ header chaining rules apply. The value in this field applies to WMQ header which follows immediately after this one, and the corresponding field in the final WMQ header before the payload MUST be set to the value shown in this table.

2.8.3.2 Behaviour of Receiving SOAP Node

When a message is successfully received by a SOAP node, that node MUST populate `mep:InboundMessage` with the received message and a transition to `Success` is made.

If the Receiving SOAP Node fails to receive a message or a message cannot be understood, then the failure reason MUST be set appropriately and a transition to `Fail` is made.

Success and Fail

`Success` and `Fail` are the terminal states for the One-Way MEP. From the point-of-view of the local node this message exchange has completed.

2.9 Faults

The fault subcodes which are listed parenthetically in this document apply only to SOAP exchanges, and are listed below:

- `wmqservice:mismatchedSoapAction`
- `wmqservice:mismatchedContentType`
- `wmqservice:unrecognizedBindingVersion`
- `wmqservice:malformedFormat`
- `wmqservice:malformedMsgId`
- `wmqservice:malformedCorrelId`
- `wmqservice:encodingNotAllowedForSoapBindings`
- `wmqservice:targetActionNotAllowedForSoapBindings`
- `wmqservice:unknownCharacterSet`

[SOAP 1.2 Part 1] specifies that the a SOAP fault MUST consist of a Code and Reason Element. All of the subcodes listed above apply to the `Sender` SOAP fault code defined in [SOAP 1.2 Part 1]. The fault `subcode` element MAY be included as a child of the `Code` element, and if it is included, it MUST have a child called `Value` which contains the actual subcode. Service providers are free to provide their own descriptive text in the `Reason` element.

The following example shows a SOAP 1.2 fault with a `wmqservice:mismatchedSoapAction` subcode:

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
            xmlns:wmqservice="http://www.ibm.com/xmlns/prod/wmq/bindings/1.0"
            xmlns:xml="http://www.w3.org/XML/1998/namespace">
 <env:Body>
  <env:Fault>
   <env:Code>
     <env:Value>env:Sender</env:Value>
     <env:Subcode>
      <env:Value>wmqservice:mismatchedSoapAction</env:Value>
     </env:Subcode>
   </env:Code>
   <env:Reason>
     <env:Text xml:lang="en">The action parameter in the content type does not
match the soapAction property in the MQRFH2.</env:Text>
   </env:Reason>
  </env:Fault>
 </env:Body>
</env:Envelope>
```

SOAP 1.1 does not define a mechanism for carrying a fault subcode, so when issuing SOAP 1.1 faults, WMQ service requesters and providers SHOULD use the fault's `detail` element to carry the subcode. The `detail` element SHOULD include a single child element with a (namespace-qualified) name which matches the subcodes. A service requester or provider MAY put a textual description as the value of the `detail` element to provide more information about the error.

The following example shows a SOAP 1.1 fault for a `wmqservice:mismatchedSoapAction` subcode with additional text in the `detail` element.

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
   <env:Body>
      <env:Fault>
        <faultcode>env:Client</faultcode>
        <faultstring>Client Error</faultstring>
        <detail>
           <wmqservice:mismatchedSoapAction
             xmlns:wmqservice="http://www.ibm.com/xmlns/prod/wmq/bindings/1.0">
              The action parameter in the content type does not match
              the soapAction property in the MQRFH2
            </wmqservice:mismatchedSoapAction>
        </detail>
      </env:Fault>
   </env:Body>
</env:Envelope>
```

2.10 Examples

2.10.1 MEP Example 1: Native WMQ One-Way

2.10.1.1 One-Way IRI

The following IRI describes the address of a native WMQ one-way application. The application is waiting for binary messages to arrive at the INS.ACCIDENT.REPORT queue on queue manager MOTOR.INS. The application expects persistent messages with the ACCREP format..

```
wmq:/msg/queue/INS.ACCIDENT.REPORT?connectQueueManager=MOTOR.INS
       &persistence=MQPER_PERSISTENT&format=ACCREP
```

In the absence of any other information (e.g. from a WSDL document) the one-way MEP can be inferred from the lack of a replyTo destination as described in section 2.8.1.1 - Determining the MEP. This MEP could be made more explicit by including a msgType=MQMT_DATAGRAM parameter. Section 3.3.4 - WSDL Example 4 – Native WMQ One-way shows a WSDL document describing this service.

2.10.1.2 One-Way Message

The following represents a human readable version of a one-way WMQ datagram message which could be sent to this application. The service definition does not contain any data that needs to be carried in an MQRFH2 header, so no MQRFH2 is added.

```
[ 364 bytes] Message Descriptor (MQMD)
StrucId      :'MD  '
Version      :2
Report       :0
Message Type :8
Expiry       :-1
Feedback     :0
MQEncoding   :0x'222'
CCSID        :850
Format       :'ACCREP  '
Priority     :0
Persistence  :1 (Persistent)
Message Id   :A M Q   M O T O R . I N S       . 9 d E   . . .
```

```
               414D51204D4F544F522E494E53202020AF39644520045F17
Correl. Id    :. . . . . . . . . . . . . . . . . . . . . . . .
               000000000000000000000000000000000000000000000000
Backout Cnt. :0
ReplyToQ      :'                                                 '
ReplyToQMgr   :'MOTOR.INS                                        '
UserId        :'Administrato'
AccountingTkn:. . . . . . . . e t A d ; . k b . . 9 . . . . . . . . . . . . . .
               16010515000000B8657441643BB26B6200A039F40100000000000000000000000B
ApplIndentity:'                                  '
PutApplType   :11
PutApplName   :'WebSphere MQ\bin\invoke.exe'
Put Date      :'20061122'
Put Time      :'19055659'
ApplOriginDat:'    '
Group Id      :. . . . . . . . . . . . . . . . . . . . . . . .
               000000000000000000000000000000000000000000000000
Msg Seq No.   :1
Offset        :0
MsgFlags      :0
Original Len.:-1
+00000000 00C0FFEE 00C0FFEE FEEDFEED F00DF00D           ................
```

2.10.2 MEP Example 2: Native WMQ Request-Response Pair

2.10.2.1 Request-Response IRI

The following IRI describes a native WMQ request/response application designed to process an insurance quote request and place a response on a reply queue. The presence of the replyTo parameter tells the consumer of this service to expect a response.

```
wmq:/msg/queue/INS.QUOTE.REQUEST?connectQueueManager=MOTOR.INS
       &replyTo=msg/queue/INS.QUOTE.REPLY
       &format=MQSTR
       &persistence=MQPER_NOT_PERSISTENT
       &reportOptions=MQRO_PASS_MSG_ID
```

The request-response MEP could also be indicated by including a msgType=MQMT_REQUEST parameter, in which case the replyTo parameter could be omitted from the IRI and the service requester would be expected to provide the replyTo destination.

The `format` and `persistence` parameters specify that the messages are non-persistent text messages.

The `reportOptions` parameter specifies tells the responding application to set the msgId of the response to the msgId of the request message in addition to the default behaviour of copying the request msgId to the correlId of the response.

2.10.2.2 Native WMQ Request Message

The following represents a human readable version of the WMQ request message. Note that the service definition does not contain any data that needs to be carried in an MQRFH2 header, so no MQRFH2 is added. Section 3.3.5 - WSDL Example 5 - Native WMQ Request-Response shows a WSDL document describing this service.

```
[  364 bytes] Message Descriptor (MQMD)
StrucId      :'MD  '
Version      :2
Report       :128
Message Type :1
Expiry       :-1
Feedback     :0
MQEncoding   :0x'222'
CCSID        :850
Format       :'MQSTR   '
Priority     :0
Persistence  :0 (Not Persistent)
Message Id   :A M Q   M O T O R . I N S       . 9 d E   . . .
              414D51204D4F544F522E494E53202020AF39644520000F04
Correl. Id   :. . . . . . . . . . . . . . . . . . . . . . . .
              000000000000000000000000000000000000000000000000
Backout Cnt. :0
ReplyToQ     :'INS.QUOTE.REPLY                                 '
ReplyToQMgr  :'MOTOR.INS                                       '
UserId       :'Administrato'
AccountingTkn:. . . . . . . . e t A d ; . k b . . 9 . . . . . . . . . . . . . .
              16010515000000B8657441643BB26B6200A039F4010000000000000000000000B
ApplIndentity:'                                '
PutApplType  :11
PutApplName  :'WebSphere MQ\bin\invoke.exe'
Put Date     :'20061122'
Put Time     :'19055659'
ApplOriginDat:'    '
Group Id     :. . . . . . . . . . . . . . . . . . . . . . . .
              000000000000000000000000000000000000000000000000
Msg Seq No.  :1
Offset       :0
MsgFlags     :0
Original Len.:-1
+00000000 4A616775 61722C58 4A31322C 31393834 2C313230  Jaguar,XJ12,1984,120
+00000014 3030302C 31323030 2C33392C 73747265 65742070  000,1200,39,street p
+00000028 61726B69 6E672C33                             arking,3
```

2.10.2.3 Native WMQ Response Message

```
[  364 bytes] Message Descriptor (MQMD)
StrucId      :'MD  '
Version      :2
Report       :0
Message Type :2
Expiry       :-1
Feedback     :0
MQEncoding   :0x'222'
CCSID        :850
Format       :'MQSTR   '
Priority     :0
Persistence  :0 (Not Persistent)
Message Id   :A M Q   M O T O R . I N S       . 9 d E   . . .
              414D51204D4F544F522E494E53202020AF39644520000F04
Correl. Id   :A M Q   M O T O R . I N S       . 9 d E   . . .
              414D51204D4F544F522E494E53202020AF39644520000F04
Backout Cnt. :0
ReplyToQ     :'                                                '
ReplyToQMgr  :'MOTOR.INS                                       '
UserId       :'Administrato'
```

```
AccountingTkn:. . . . . . . . . e t A d ; . k b . . 9 . . . . . . . . . . . . . .
              16010515000000B8657441643BB26B6200A039F401000000000000000000000B
ApplIndentity:'                                 '
PutApplType  :11
PutApplName  :'WebSphere MQ\bin\respond.exe'
Put Date     :'20061122'
Put Time     :'19020579'
ApplOriginDat:'    '
Group Id     :. . . . . . . . . . . . . . . . . . . . . . . .
              000000000000000000000000000000000000000000000000
Msg Seq No.  :1
Offset       :0
MsgFlags     :0
Original Len.:-1
[ Message body ]
+00000000 596F7572 20696E73 7572616E 63652066 6F722074  Your insurance for t
+00000014 68652079 65617220 77696C6C 20626520 9C333030  he year will be £300
+00000028 20
```

2.10.3 MEP Example 3: SOAP Request-Response Exchange

2.10.3.1 SOAP Request/Response IRI

The following IRI describes another WMQ request/response service which provides insurance quotes, however this application has been deployed with a SOAP/WMQ binding. The binding is not specified in the IRI, so would need to be specified in the WSDL, or in the tools used to generate a SOAP proxy for the service requester. Section 3.2.4 - WSDL Example 2 - SOAP/WMQ Request-Response shows a WSDL document which describes this service.

```
wmq:/msg/queue/INS.QUOTE.SOAP.REQUEST?connectQueueManager=MOTOR.INS
       &replyTo=msg/queue/INS.QUOTE.SOAP.REPLY
       &usrRoutingData=UserDefinedRoute
```

2.10.3.2 SOAP Request Message

The following represents a human readable version of the SOAP request message. The message includes an MQRFH2 header because the IRI contains a user defined 'usrRoutingData' parameter. This is carried in the `<usr>` subfolder of the `<mq_svc>` MQRFH2 folder.

```

[  364 bytes] Message Descriptor (MQMD)
StrucId      :'MD  '
Version      :2
Report       :0
Message Type :1
Expiry       :-1
Feedback     :0
MQEncoding   :0x'222'
CCSID        :1208
Format       :'MQHRF2  '
Priority     :0
Persistence  :0 (Not Persistent)
Message Id   :A M Q   M O T O R . I N S       . 9 d E   . . .
              414D51204D4F544F522E494E53202020AF3964452000DE12
Correl. Id   :. . . . . . . . . . . . . . . . . . . . . . . .
              000000000000000000000000000000000000000000000000
Backout Cnt. :0
```

```
ReplyToQ      :'INS.QUOTE.SOAP.REPLY                            '
ReplyToQMgr   :'MOTOR.INS                                       '
UserId        :'Administrato'
AccountingTkn:. . . . . . . . e t A d ; . k b . . 9 . . . . . . . . . . . . .
              16010515000000B8657441643BB26B6200A039F4010000000000000000000000B
ApplIndentity:'                                '
PutApplType   :11
PutApplName   :'WebSphere MQ\bin\SOAPproxy.exe'
Put Date      :'20061109'
Put Time      :'17134957'
ApplOriginDat:'    '
Group Id      :. . . . . . . . . . . . . . . . . . . . . . . . .
              000000000000000000000000000000000000000000000000
Msg Seq No.   :1
Offset        :0
MsgFlags      :0
Original Len.:-1
[  828 bytes] Rules and Formatting (MQRFH2)
StrucId       :'RFH '
Version       :2
Struc Length :316
MQEncoding    :0x'222'
CCSID         :850
Format        :'MQSOAP11'
RFH Flags     :00000000
NameVal CCSID:1208
<mq_svc>
     <bindingVersion>1.0</bindingVersion>
     <usr>
       <RoutingData>UserDefinedRoute</RoutingData>
     </usr>
     <contentType>
       text/xml;
       charset=utf-8
     </contentType>
 </mq_svc>
[ Message body ]
<?xml version="1.0" encoding="UTF-8"?>
 <soapenv:Envelope
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body
     <getQuote xmlns="http://example.org/quotes/">
       <make>Jaguar</make>
       <model>XJ12</model>
       <year>1984</year>
       <mileage>120000</mileage>
       <value>1200</value>
     </getQuote>
   </soapenv:Body>
 </soapenv:Envelope>
```

The SOAP response message is not included in this example.

3 WSDL Usage

3.1 Overview

The preceding section described the rules by which messages are sent and received using WMQ. The following sections detail how to use WSDL to describe services sending SOAP and non-SOAP messages over WMQ. Section 3.2 describes the considerations for SOAP WSDL bindings; Section 3.3 describes the considerations for non-SOAP (native) WSDL bindings, and Section 3.4 describes how to specify WMQ service properties for both SOAP and non-SOAP WSDL bindings.

Points to note in section 3 include:

- A new WMQ native binding is defined to allow the description of WMQ applications that do not use SOAP or other WSDL binding types;
- A new WMQ transport URI value is defined for the transport attribute of the `soap:binding` element
- The use of the soapAction header is allowed, even though it is explicitly disallowed for non-HTTP transports by the WSDL specification.
- These sections define how to set the `wmqservice` properties in WSDL to control the connection and runtime behaviours of the binding.

3.2 WSDL 1.1 Extensions for WMQ SOAP Bindings

This section describes the WSDL extensions which are needed to describe WMQ as a transport for SOAP. For general information on extending SOAP bindings in WSDL, please refer to the SOAP Binding from [WSDL 1.1]. For information about SOAP 1.2 bindings, see [WSDL 1.1 Extension for SOAP 1.2] .

3.2.1 Transport Identification

The `wsdl11soap11:binding` and `wsdl11soap12:binding` elements have a `transport` attribute. When using WMQ as the transport, this value MUST be set to `"http://www.ibm.com/xmlns/prod/wmq/transport"`. For example:

```
<soap:binding style="document"
transport="http://www.ibm.com/xmlns/prod/wmq/transport"/>
```

3.2.2 SOAP Action

The wsdl11soap11:operation portion of the WSDL specification explicitly disallows use of the `soapAction` attribute in non-HTTP bindings. This specification supersedes that requirement, and allows the use of `soapAction` in the wsdl11soap11:operation element for SOAP/WMQ bindings. This value corresponds to the property `wmqservice:soapAction`.

3.2.3 WSDL Example 1 – WMQ and HTTP SOAP Bindings

This first example of a WMQ SOAP binding is an adaptation of the sample service definition included in the "Example 1 SOAP 1.1 Request/Response via HTTP" section of [WSDL 1.1] (see http://www.w3.org/TR/wsdl#_wsd). The original service (in [WSDL 1.1]) is available using HTTP. This example shows how the service may also be made available over WMQ.

The following WSDL extract includes the HTTP `<binding>` (which is unmodified) in lines 1-12, a new WMQ `<binding>` in lines 14-29, and a new WMQ `<port>` definition in lines 36-38, which is included alongside the HTTP `<port>` in the `<service>` element.

```
1     <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
2        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
3         <operation name="GetLastTradePrice">
4            <soap:operation
soapAction="http://example.com/GetLastTradePrice"/>
5           <input>
6               <soap:body use="literal"/>
7            </input>
8           <output>
9               <soap:body use="literal"/>
10          </output>
11        </operation>
12    </binding>
13
14   <binding name="StockQuoteSoapBinding_wmq" type="tns:StockQuotePortType"
xmlns:wmqservice="http://www.ibm.com/xmlns/prod/wmq/bindings/1.0">
15       <soap:binding style="document"
transport="http://www.ibm.com/xmlns/prod/wmq/transport"/>
16
17       <!-- Specify PERSISTENT delivery mode -->
18       <wmqservice:persistence>MQPER_PERSISTENT</wmqservice:persistence>
19
20       <operation name="GetLastTradePrice">
21         <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
22         <input>
23             <soap:body use="literal"/>
24         </input>
25         <output>
26             <soap:body use="literal"/>
27          </output>
28       </operation>
29   </binding>
30
31   <service name="StockQuoteService">
32       <documentation>My first service</documentation>
33       <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
34           <soap:address location="http://example.com/stockquote"/>
35       </port>
36       <port name="StockQuotePort_wmq"
binding="tns:StockQuoteSoapBinding_wmq">
37           <soap:address location="wmq:/msg/queue/STOCKQUOTE/>
38       </port>
39   </service>
```

The key points to notice in this WSDL fragment are:

- The WMQ transport URI in `<soap:binding>` (line 15)
- The wmq: queue location IRI in the `<soap:address>` (line 37)
- The `persistence` extension property in the `<soap:binding>` (line 18)

3.2.4 WSDL Example 2 - SOAP/WMQ Request-Response

The following WSDL example shows a WSDL document which corresponds to the SOAP request-response application described in section 2.10.3 - MEP Example 3: SOAP Request-Response Exchange. In contrast to the earlier MEP example, the `wmqservice` properties are not specified exclusively in the location IRI. They have been placed in the WSDL in the positions which are most relevant to the elements they apply to.

```
 1. <definitions ...
       xmlns:wmqservice="http://www.ibm.com/xmlns/prod/wmq/bindings/1.0">
 2.  <!-- Standard WSDL message and portType definitions go here -->
 3.  <!- Binding: specifies persistence, message formats, and correlation
       style -->
 4.   <binding name="mq_insurance_bindings" type="tns:portType1">
 5.     <soap:binding style="document"
       transport="http://www.ibm.com/xmlns/prod/wmq/transport"/>
 6.     <wmqservice:usr><RoutingData>UserDefinedRoute</RoutingData></wmqservi
       ce:usr>
 7.     <operation name="getQuote">
 8.         <input>
 9.           <soap:body use="literal"/>
10.         </input>
11.         <output>
12.           <soap:body use="literal"/>
13.         </output>
14.     </operation>
15.   </binding>
16.   <service name="InsuranceServices">
17.     <port name="mq_ports" binding="tns:mq_insurance_bindings">
18.       <soap:address location="wmq:/msg/queue/INS.QUOTE.SOAP.REQUEST"/>
19.       <wmqservice:replyTo>msg/queue/INS.QUOTE.SOAP.REPLY</wmqservice:repl
       yTo>
20.       <wmqservice:connectQueueManager>MOTOR.INS</wmqservice:connectQueueM
       anager>
21.     </port>
22.   </service>
23.</definitions>
```

The key points to notice in this WSDL fragment are:

- The WMQ transport URI in the `<soap:binding>` (line 5)
- The user defined `wmqservice:usr/RoutingData` property in the `<port>` (line 6)
- The `wmq:` queue location IRI in the `<soap:address>` (line 18)
- The `wmqservice:replyTo` extension property in the `<port>` (line 19)
- The WMQ `wmqservice:connectQueueManager` extension property in the `<port>` (line 20)

3.3 WSDL 1.1 Extensions for Native WMQ Bindings

This section describes the native (non-SOAP) WMQ WSDL binding. The native WSDL binding allows developers to create a WSDL description for WMQ applications that do not use SOAP messages and so cannot be described using the SOAP WSDL binding. The WMQ native binding supports the specification of the following protocol specific information:

- An indication that a binding MUST use the WMQ protocol
- A way of specifying an address for a WMQ endpoint.
- A list of the WMQ service properties which apply to the service.

3.3.1 WMQ Native Binding WSDL Extensions

The WMQ Native Binding extends WSDL with the following extension elements:

```
<wmqservice:binding/>
<wmqservice:body/>
<wmqservice:address location="xsd:anyURI">
```

These elements are detailed below.

wmqservice:binding
This element is used to indicate that this binding uses the native WMQ transport
The `wmqservice:binding` element MUST be present when using the WMQ native binding.

- This element can only be used in the
  `wsdl:definitions/wsdl:binding/wmqservice:binding`
  location in a WSDL document.

wmqservice:body
The `wmqservice:body` element specifies how the message appears as the WMQ message body.

- This element can only be used in the:
  `wsdl:definitions/wsdl:binding/wsdl:operation/wsdl:input/wmqservice:body`
  and/or
  `wsdl:definitions/wsdl:binding/wsdl:operation/wsdl:output/wmqservice:body`
  and/or
  `wsdl:definitions/wsdl:binding/wsdl:operation/wsdl:fault/wmqservice:body`
  locations in a WSDL document.

The `wmqservice:body` corresponds to a message (input, output, or fault) in the `wsdl:portType/wsdl:operation` which in turn refers to a WSDL message. The corresponding WSDL message SHOULD have zero or one part. If the WSDL message does contain more than one part, only the first part SHOULD be used. If the message has no parts then an empty message SHOULD be sent.

wmqservice:address
The `wmqservice:address` element defines the WMQ endpoint to which the message SHOULD be sent.

- This element can only be used in the
  `wsdl:definitions/wsdl:service/wsdl:port/wmqservice:address`
  location in a WSDL document.

In addition to these elements, the WMQ service definition properties (detailed in section 2 of this specification) can be used in WSDL to describe WMQ applications (see section 3.4 - Specifying WMQ Properties in WSDL for more details).

Section 3.3.4 - WSDL Example 4 – Native WMQ One-way contains an example which shows how all these elements are used in a WSDL fragment.

3.3.2 Describing WMQ Native Message Formats

The WMQ native binding does not extend or change WSDL's intrinsic XML Schema-based type system, so all messages that are to be sent in the native WMQ binding have to be modelled in XML Schema. This modelling is reasonably straightforward for XML messages - if the message is a simple type it can be modelled as an xsd:type, but otherwise it MUST be modelled as an xsd:element.

Modelling is more problematic for WMQ messages encoded in non-XML data formats. XML Schema is an extensible system and so *can* be used to model non-XML messages, such as 'C' structures and COBOL copy-books, but basic XML schema has no standard extensions for defining wire representations of data (for example the encoding of numeric data). Schema also has only limited abilities for describing other non-XML data like 'Tagged-Delimited' messages. To overcome these limitations non-XML data is commonly described in XML schema which has been extended by proprietary annotations.

WSDL allows the use of these proprietary schema annotations, so their use is permitted in a WMQ Service definition, but developers should be aware that using proprietary annotations may limit the ability to port a WMQ service definition between different vendor's tools.

A set of schema annotations and rules - called "Data Format Description Language" (DFDL) - is being developed to extend XML schema to describe non-XML data formats. The [Data Format Description Language WG] specification is under development at the time of writing, but once finalised, DFDL will provide a useful standard for describing native non-XML WMQ messages that cannot be described in basic XML schema .

3.3.3 WSDL Example 3 - WMQ Native Binding

The following is a simple - but complete - WSDL example that shows the binding of a request-response application that takes an MQSTR format as persistent input and output messages. No other WMQ service properties are specified, so no headers other than the default MQMD need to be used. No replyTo destination is specified, so the caller would be expected to provide a replyTo destination.

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions
  targetNamespace="http://www.example.org/wmq/wsdl"
  xmlns:tns="http://www.example.org/wmq/wsdl"
  xmlns:wmqservice="http://www.ibm.com/xmlns/prod/wmq/bindings/1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  name="MQExampleService" >

  <wsdl:message name="m1">
    <wsdl:part name="part1" type="xsd:string" />
  </wsdl:message>

  <wsdl:message name="m2">
```

```
15.     <wsdl:part name="part1" type="xsd:string" />
16.   </wsdl:message>
17.
18.   <wsdl:portType name="pt1">
19.     <wsdl:operation name="o1">
20.       <wsdl:input message="tns:m1" />
21.       <wsdl:output message="tns:m2" />
22.     </wsdl:operation>
23.   </wsdl:portType>
24.
25.   <wsdl:binding name="b1" type="tns:pt1">
26.     <wmqservice:binding/>
27.     <!-- Specify PERSISTENT delivery mode -->
28.     <wmqservice:persistence>MQPER_PERSISTENT</wmqservice:persistence>
29.     <wmqservice:format>MQSTR</wmqservice:format>
30.     <wsdl:operation name="o1">
31.       <wsdl:input>
32.         <wmqservice:body />
33.       </wsdl:input>
34.       <wsdl:output>
35.         <wmqservice:body />
36.       </wsdl:output>
37.     </wsdl:operation>
38.   </wsdl:binding>
39.
40.   <wsdl:service name="service1">
41.     <wsdl:port name="port1" binding="tns:b1">
42.       <wmqservice:address location="wmq:/msg/queue/Q1" />
43.     </wsdl:port>
44.   </wsdl:service>
45.</wsdl:definitions>
```

The key points to notice in this WSDL document are:

- The `wmq:` queue location IRI in the `<wmqservice:address>` (line 42)
- The `<wmqservice:binding/>` element in the WSDL `<binding>` which indicates that WMQ MUST be used as the binding and transport (line 26)
- The `wmqservice:persistence` extension property in the `<binding>` (line 28)
- The WMQ `wmqservice:format` extension property in the `<binding>` (line 29) to specify that text message formats MUST be used.
- The `<wmqservice:body/>` elements (line 32 and 35) to identify the input and output messages as WMQ messages.

3.3.4 WSDL Example 4 – Native WMQ One-way

The following WSDL example shows a WSDL document which corresponds to the native one-way application described in section 2.10.1 - MEP Example 1: Native WMQ One-Way. In contrast to the earlier MEP example, the `wmqservice` properties are not specified exclusively in the location IRI. They have been placed in the WSDL in positions which are most relevant to the elements they apply to.

```
 1. <definitions ...
       xmlns:wmqservice="http://www.ibm.com/xmlns/prod/wmq/bindings/1.0">
 2.  <!-- WSDL message and portType definitions (WMB or DFDL schema
       annotations) -->
 3.   <! - Binding - specifies persistence -->
 4.  <binding name="mq_accident_report" type="tns:portType1">
 5.     <wmqservice:binding/>
 6.     <wmqservice:persistence>MQPER_PERSISTENT</wmqservice:persistence>
 7.     <operation name="sendReport">
 8.        <! - Unlike the IRI  the message format is specified on the
       message binding -->
 9.        <input>
10.          <wmqservice:body/>
11.          <wmqservice:format>ACCREP</wmqservice:format>
12.        </input>
13.     </operation>
14.   </binding>
15.   <! - Standard Service def. -->
16.   <service name="AccidentServices">
17.    <! - Port - specifies target Queue and Queue manager -->
18.     <port name="mq_ports" binding="tns:mq_accident_report">
19.       <wmqservice:address location="wmq:/msg/queue/INS.ACCIDENT.REPORT"/>
20.       <wmqservice:connectQueueManager>
          MOTOR.INS</wmqservice:connectQueueManager>
21.     </port>
22.   </service>
23.</definitions>
```

The key points to notice in this WSDL fragment are:

- The `wmq:` queue location IRI in the `<wmqservice:address>` (line 19)
- The `<wmqservice:connectQueueManager>` element in the WSDL `<port>` which tells the caller which queue manager to connect to (line 20).
- The `<wmqservice:binding/>` element in the WSDL `<binding>` which indicates that WMQ MUST be used as the binding and transport (line 5)
- The `wmqservice:persistence` extension property in the `<binding>` (line 6)
- The `<wmqservice:body/>` element (line 10) to identify the input messages as a WMQ message
- The WMQ `wmqservice:format` extension property in the operation's `<input>` message (line 11) to specify that the message format is "ACCREP".

3.3.5 WSDL Example 5 - Native WMQ Request-Response

The following WSDL example shows a WSDL document which corresponds to the native request-response application described in section 2.10.2 - MEP Example 2: Native WMQ Request-Response Pair. In contrast to the earlier MEP example, the `wmqservice` properties are not specified exclusively in the location IRI. They have been placed in the WSDL in positions which are most relevant to the elements they apply to.

```
 1. <definitions ...
        xmlns:wmqservice="http://www.ibm.com/xmlns/prod/wmq/bindings/1.0">
 2. <!-- WSDL message and portType definitions (maybe WMB or DFDL schema
       annotations) -->
 3.  <!- Binding - specifies persistence, message formats, and correlation
       style -->
 4.   <binding name="mq_insurance_bindings" type=“tns:portType1">
 5.     <wmqservice:binding/>
 6.     <wmqservice:persistence>MQPER_NOT_PERSISTENT</wmqservice:persistence>
 7.     <wmqservice:format>MQSTR</wmqservice:format>
 8.     <wmqservice:reportOptions>MQRO_PASS_MSG_ID</wmqservice:reportOptions>
 9.     <operation name=“getQuote">
10.         <input>
11.           <wmqservice:body/>
12.         </input>
13.         <output>
14.           <wmqservice:body/>
15.         </output>
16.     </operation>
17.   </binding>
18.   <service name=“InsuranceServices">
19.     <port name=“mq_ports" binding=“tns:mq_insurance_bindings">
20.       <wmqservice:address location=" wmq:/msg/queue/INS.QUOTE.REQUEST"/>
21.       <wmqservice:replyTo>msg/queue/INS.QUOTE.REPLY</wmqservice:replyTo>
22.     </port>
23.   </service>
24.</definitions>
```

The key points to notice in this WSDL fragment are:

- The `wmq:` queue location IRI in the `<wmqservice:address>` (line 20);
- The `<wmqservice:replyTo>` element in the WSDL `<port>` (line 21);
- The `<wmqservice:binding/>` element in the WSDL `<binding>` which indicates that WMQ MUST be used as the binding and transport (line 5);
- The `wmqservice:persistence` extension property in the `<binding>` (line 6);
- The WMQ `wmqservice:format` extension property in the `<binding>` (line 7) to specify that the all messages for this binding are text messages (unless overridden – see 3.4.3 - Precedence of Properties In WSDL);
- The WMQ `wmqservice:reportOptions` extension property in the `<binding>` to specify the way in which requests and responses are correlated (line 8).
- The `<wmqservice:body/>` elements (line 11 and 14) to identify the input and output messages as WMQ messages.

3.4 Specifying WMQ Properties in WSDL

Property names are always case-sensitive, wherever they are specified.

3.4.1 Properties

The following table shows which WSDL elements can be extended with WMQ service binding properties. The properties are added into the WSDL extension points for these elements.

The columns represent:

- the “location IRI” attribute in the `wsdl:port/soap:address` element, or `wsdl:port/wmqservice:address` element;
- the `wsdl:port` element; the `wsdl:service` element; the `wsdl:binding` element; the `wsdl:binding/wsdl:operation` element; the `wsdl:binding/wsdl:operation/wsdl:input` element; the `wsdl:binding/wsdl:operation/wsdl:output` element; and the `wsdl:binding/wsdl:operation/wsdl:fault` element.
- Whether the property is allowed for SOAP and/or Native WMQ bindings.

Property Name	**location IRI**	**port**	**service**	**binding**	**operation**	**input**	**output**	**fault**	**SOAP/ Native**
destinationName	Y	N	N	N	N	N	N	N	Both
connectQueueManager	Y	Y	N	N	N	N	N	N	Both
channelTableName	Y	Y	N	N	N	N	N	N	Both
channelTableLib	Y	Y	N	N	N	N	N	N	Both
channelName	Y	Y	N	N	N	N	N	N	Both
connectionName	Y	Y	N	N	N	N	N	N	Both
transportType	Y	Y	N	N	N	N	N	N	Both
replyTo	Y	Y	Y	Y	Y	Y	Y	Y	Both
targetAction	Y	N	N	N	Y	N	N	N	Native
soapAction	Y	N	N	N	Y	N	N	N	SOAP
contentType	Y	Y	Y	Y	Y	Y	Y	Y	SOAP
reportOptions	Y	Y	Y	Y	Y	Y	Y	Y	Both
msgType	Y	Y	Y	Y	Y	Y	Y	Y	Both
expiry	Y	Y	Y	Y	Y	Y	Y	Y	Both
priority	Y	Y	Y	Y	Y	Y	Y	Y	Both
persistence	Y	Y	Y	Y	Y	Y	Y	Y	Both
msgId	Y	Y	Y	Y	Y	Y	Y	Y	Both
correlId	Y	Y	Y	Y	Y	Y	Y	Y	Both
codedCharSetId	Y	Y	Y	Y	Y	Y	Y	Y	Both
usr	Y	Y	Y	Y	Y	Y	Y	Y	Both
format	Y	Y	Y	Y	Y	Y	Y	Y	Native
encoding	Y	Y	Y	Y	Y	Y	Y	Y	Native

3.4.2 Specifying Properties Via the WMQ IRI

Most of the `wmqservice` properties can be put in the location IRI (as discussed earlier in this specification and defined in [WMQ-IRI]). The WSDL IRI, is set as the location attribute of the `<soap:address>` or `<wmqservice:address>` element in the WSDL port.

When properties are defined in the IRI, the property name MUST NOT be qualified with the `wmqservice` namespace prefix - the simple property name such as `priority` MUST be used.

If a property is specified more than once on the WMQ IRI the last instance of the property will be used.

3.4.3 Precedence of Properties In WSDL

The WMQ properties described in this specification may be set in a number of places in the WSDL (see section 3.4.1 - Properties). If a property is specified more than once in the WSDL, the service requester determines which value to use by aggregating the `wmqservice` properties in the WSDL elements into a set of "effective properties".

To simplify the aggregation, the properties are grouped together into a number of "effective property scopes". In the event of a property being specified in multiple places, the most significant property MUST be used. These effective scopes listed in order of significance are:

- Effective Service Property Scope (least significant);
- Effective Endpoint Property Scope;
- Effective Operation Property Scope;
- Effective Input Property Scope (most significant);
- Effective Output Property Scope (most significant);
- Effective Fault Property Scope (most significant).

WMQ property scopes and effective policies are defined below.

3.4.3.1 Effective Service properties

The *effective properties* for a service are the properties specified on the Service – i.e.

- wsdl:service.

3.4.3.2 Effective Endpoint Properties

The *effective properties* for an Endpoint are the combined properties (in order from least to most significant) for the Service, the Port, the location address IRI in the Port, and the Binding associated with the Port – i.e.

- Effective Service properties ;
- AND wsdl:port properties ;
- AND wsdl:port address location IRI attribute properties;
- AND wsdl:binding properties;

3.4.3.3 Effective Operation Properties

The *effective properties* for an Operation are the combined properties (in order from least to most significant) for the Service, the Endpoint and the operation for the binding – i.e. :

- Effective Service Properties ;
- AND Effective Endpoint Properties ;
- AND wsdl:binding/wsdl:operation.

3.4.3.4 Effective Input Properties

The *effective properties* for an Input Message are the combined properties (in order from least to most significant) for the Service, the Endpoint, the Operation, and the binding operation's Input message – i.e.:

- Effective Service Properties;

- AND Effective Endpoint Properties;
- AND Effective Operation Properties;
- AND wsdl:binding/wsdl:operation/wsdl:input.

3.4.3.5 Effective Output Properties

The *effective properties* for an Output Message are the combined properties (in order from least to most significant) for the Service, the Endpoint, the Operation, and the binding operation's Output message – i.e.:

- Effective Service Properties;
- AND Effective Endpoint Properties;
- AND Effective Operation Properties;
- AND wsdl:binding/wsdl:operation/wsdl:output.

3.4.3.6 Effective Fault Properties

The *effective properties* for an Fault Message are the combined properties (in order from least to most significant) for the Service, the Endpoint, the Operation, and the appropriate Fault message from the binding operation – i.e.:

- Effective Service Properties;
- Effective Endpoint Properties;
- Effective Operation Properties;
- wsdl:binding/wsdl:operation/wsdl:fault.

For example, if the `wmqservice:persistence` property is defined for a `wsdl:service` element, then the effective `wmqservice:persistence` property value at all scopes will be taken from the `wsdl:service` element, unless it is overridden by a `wmqservice:persistence` property at any other scope level.

In the following example, notice the `wmqservice:expiry` property. For the "quickPort" port, the value will be taken from the port level and will be 10 (one second). For the "slowPort" port, the value will be taken from the service level and will be will be 100 (10 seconds).   For both ports the messages will be non-persistent.

```
<wsdl:binding name="exampleBinding">
  ...
</wsdl:binding>
<wsdl:service name="exampleService">
  <wmqservice:persistence>MQPER_NOT_PERSISTENT</wmqservice:persistence>
  <wmqservice:expiry>100</wmqservice:expiry>
  ...
  <wsdl:port name="quickPort" binding="tns:exampleBinding">
    ...
    <wmqservice:expiry>10</wmqservice:expiry>
  </wsdl:port>
  <wsdl:port name="slowPort" binding="tns:exampleBinding">
    ...
  </wsdl:port>
</wsdl:service>
```

It is the service requester's or provider's responsibility to determine the appropriate property scope for the action it is taking. For example, when connecting to a queue manager to send an input message to a service's operation it uses the "Input Message" property scope; When connecting to a queue manager for some other activity (which is not directly associated with a operation message) it uses “Endpoint” property scope.

4 Appendix A: References

SOAP-JMS

SOAP Bindings for Java Message Service (JMS) , To be published. (See http://www.soapjms.org/2007/08/soap/bindings/JMS/)

RFC 2045

IETF *RFC 2045 Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies* , N. Freed, Innosoft, N. Borenstein, 1996. (See http://www.ietf.org/rfc/rfc2045.txt)

RFC 2046

IETF RFC 2046 Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types N. Freed, Innosoft, N. Borenstein, 1996. (See http://www.ietf.org/rfc/rfc2046.txt)

SwA

W3C Note *SOAP Messages with Attachments* , John Barton, Satish Thatte, Henrik Frystyk Nielsen, December, 2000 (See http://www.w3.org/TR/SOAP-attachments)

SOAP11-MTOM

W3C Member Submission, *SOAP 1.1 Binding for MTOM 1.0* , Christopher Ferris, Jonathan Marsh, et. al., 05 April 2006. (See http://www.w3.org/Submission/2006/SUBM-soap11mtom10-20060405/)

SOAP12-MTOM

W3C Recommendation, *SOAP Message Transmission Optimization* , Martin Gudgin, Noah Mendelsohn, Mark Nottingham, Hervé Ruellan, 25 January 2005. (See http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/)

XML 1.0

W3C Recommendation, *Extensible Markup Language (XML) 1.0 (Fourth Edition)* (See http://www.w3.org/TR/2006/REC-xml-20060816/)

IETF RFC 2119

IETF (Internet Engineering Task Force). *RFC 2119: Key words for use in RFCs to Indicate Requirement Levels* . Scott Bradner, 1997. (See http://www.ietf.org/rfc/rfc2119.txt)

SOAP 1.1

W3C Note, *Simple Object Access Protocol (SOAP) 1.1* , Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, Dave Winer, 8 May 2000 (See http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)

SOAP 1.2 Part 1

W3C Recommendation, *SOAP Version 1.2 Part 1: Messaging Framework*, Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, Yves Lafon, 27 April 2003. (See http://www.w3.org/TR/soap12-part1/)

SOAP 1.2 Part 2

W3C Recommendation, *SOAP Version 1.2 Part 2: Adjuncts* , Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, 24 June 2003. (See http://www.w3.org/TR/soap12-part2/)

SOAP 1.2 Part 3

W3C Working Group Note, *SOAP 1.2 Part 3: One-Way MEP*, David Orchard, 2 July 2007. (See http://www.w3.org/TR/soap12-part3/)

WSDL 1.1

W3C Note, *Web Services Description Language (WSDL) 1.1* , Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, 15 March 2001. (See http://www.w3.org/TR/wsdl)

WSDL 1.1 Extension for SOAP 1.2

WSDL 1.1 Binding Extension for SOAP 1.2 , Christopher Ferris, Jonathan Marsh, et. al., 2 March 2006. (See http://schemas.xmlsoap.org/wsdl/soap12/WSDL11SOAP12.pdf)

Data Format Description Language WG

Data Format Description Language WG (See http://forge.gridforum.org/sf/projects/dfdl-wg)

SOAP Version 1.2 Email Binding

W3C Note, SOAP Version 1.2 Email Binding, Highland Mary Mountain, Jacek Kopecky, Stuart Williams, Glen Daniels, Noah Mendelsohn, 3 July 2002. (See http://www.w3.org/TR/soap12-email)

WMQ-IRI

Phillips, M, and Golby-Kirk, M, “WebSphere MQ IRI Specification”, 2007. (See WebSphere MQ SupportPac MA93).

5 Appendix B: Acknowledgements

The authors gratefully acknowledge the contributions of: Roland Merrick (IBM); Peter Niblett (IBM); Jerry Stevens (IBM); and Stephen Todd (IBM).