Version 6.1.0

MessageWay Service Interface API

Document History

Part Number	Product Name	Date
MW500-560	MessageWay Service Interface	12/2010
MW550-560	Same as above	07/2011
MW600-560	Same as above	11/2011
MW610-560	Same as above	11/2012
MW610-560	Same as above	05/2015

Copyright

This document was published on Thursday, May 14, 2015 at 13:24.

MessageWay Service Interface API

Contents

Introduction

This document defines a set of services to be provided by MessageWay through an automated mechanism. This definition assumes a request/response message passing protocol based on XML. Each service is defined as a request/response pair of XML documents.

A simple JAVA API is defined for invoking these services. This API provides a method for invoking MessageWay services by passing request XML document and receiving a response XML document. Interaction with MessageWay involves connecting to MessageWay, invoking one or more requests and then disconnecting from MessageWay.

The remainder of this document contains the following sections:

- *Interface Abstract Definition* – Definitions of all services and fields used in request/response documents
- *HTTP Interface* – Description of HTTP interface used by Java API to communicate with MessageWay.
- *DTD* – formal description of requests/responses.

Authentication and Authorization

All requests are authenticated using one of many supported authentication mechanisms.

IMPORTANT: The order of the parameters is not important, because the MessageWay Service Interface (mwsi) does not validate the order of the list. The DTDs are for reference, but because DTD does not support unordered lists, this DTD cannot be used for validation.

Request Parameters

The following request parameters are used for authentication and authorization:

Request parameters	Description
UserId	Identity to be authenticated.
AuthMethod	Authentication method. See below.
Password	MessageWay password. This is only used if *AuthMethod* is blank. The password field contains the MD5 hash of the password in ASCII hexadecimal.

Request parameters	Description
Response	Authentication credentials provided by the client to the server. See below.
Authzid	Authorization identify. This is a MessageWay user for which access rights will be granted for the remainder of the session. If blank, the the value from *User* is used.
AccessClass	Customer defined value used to control which methods a user may use to connect to MessageWay. This is typically set in the access server (FTP, SFTP, User Server, Web server, etc.) configuration. If missing, the user must be configured to allow ANY access class for the logon to succeed.

Response Parameters

The following response parameters are used for authentication and authorization:

Response parameters	Description
Challenge	Challenge data from the server. **NOTE:** This field is not currently used.

Authentication Methods

The MessageWay Service Interface supports multiple authentication methods. The logon protocol varies depending on which method is used.

Authentication Method	Description
basic	Basic authentication. The *Response* field contains the base64 encoding of *<user>*:*<password>*. This method should be avoided if the client is on a different host and HTTPS is not used.
external	External authentication. Authentication is performed by the TLS/SSL protocol. The client provides a certificate which is validated by mwsi. In this case *User* must be the common name for the client certificate. This is typically used in conjunction with *Authzid*.

Authentication Method	Description
session	Session authentication. Used for re-authentication when the client is unable to keep the connection alive during a logical session. Similar to 'basic', but uses the *SessionId* from the previously authenticated connection as credentials.
digest-md5	Digest authentication. **NOTE:** Not yet implemented.

All requests are authenticated based on the MessageWay user and password. Authentication is required only one per connection. Optional *UserId*, *Password* and *AccessClass* fields are provided on each request document for this purpose.

Version

This document describes the MessageWay Service Interface behavior when the *Version* attribute is set to **5**. The *Version* supported by the mwsi server is returned by the *Version* service.

Each service request contains an attribute providing the interface version. If this value is greater than the maximum version supported by the server, the server will effectively reduce this value to the maximum it supports. The *Logon* service may be used to negotiate the interface version. The client provides the desired version in *LoginReq* and the server returns the negotiated version, which may be less but not greater than the requested version, in *LogonRsp*. The client should thereafter use the negotiated version in subsequent requests.

Interface Abstract Definition

This section defines the interface services and request/response fields. The actual syntax of the request/response document is defined later as XML DTD.

MessageWay Service Definitions

This is a list of services and their fields or parameters.

IMPORTANT: The order of the parameters is not important, because the MessageWay Service Interface (mwsi) does not validate the order of the list. The DTDs are for reference, but because DTD does not support unordered lists, this DTD cannot be used for validation.

Logon

This service authenticates and then authorizes the user to access MessageWay.

One of several authentication methods may be used, but if the user is marked for LDAP authentication, the method here must be **basic**. All users must have a user record in MessageWay which is used to set *DefaultMailbox*, *DefaultRecipient*, and *UserRights*. Access will be denied if the user fails to authenticate or if the user is not configured to allow the supplied *AccessClass*. The success of this service starts a logical session identified by *SessionId*. The logical session should be terminated by the client by invoking the *Logoff* service.

LogonReq

The following table explains the parameters for this service.

Parameter	Description
User	Name of Identify to be authenticated.
AuthMethod	Authentication method. One of the following: ▪ **basic** ▪ **external** ▪ **session** ▪ *blank* If **basic**, then the response field contains the base64 encoding of user:password. If **external**, then authentication is by SSL client certificate and the *User* field must match the CN of the certificate. If **session**, then a previous authenticated session will be resumed. In this case *Response* should contain the *SessionId*. Other authentication methods may be added in the future.
Password	Password or password hash (MD5).
Response	Authentication credentials. The exact form depends on *AuthMethod*.
Authzid	Authorization ID. The MessageWay user for which access rights will be granted. If this is blank, then it will default to the *User* parameter.
NewPassword	Base64 encoding of <*user*>:<*new-password*>. Applicable only if the user is not configured for LDAP authentication. If present and valid, the user's MessageWay password will be changed after successful authentication.
AccessClass	Customer-defined value used to control which methods a user may use to connect to MessageWay. This is typically set in the access server (FTP, SFTP, User Server, Web server, etc.) configuration. If missing, the user must be configured to allow ANY access class for the logon to succeed.
SourceIP	Client source IP address. Perimeter servers should provide this field with the client's IP address. If not present, the session will be logged with the source IP address from the TCP connection.
PerimeterType	String indicating the type of perimeter server. Initially one of: **FTP**, **SFTP**, **WEB**, or **AS2**.
PerimeterName	Hostname of perimeter server.
AutoLogoff	If set to **True**, then the session automatically terminates when the connection is broken. Otherwise, the *Logoff* service must be called, or the session terminates after the *Idle Timeout* configured on User Policies window has expired.

LogonRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Immediate status of the message: **Success** or **Fail**.
Challenge	Server challenge. The use depends on *AuthMethod*.
SessionId	Session identifier. This is a unique cryptographically secure random value.
DefaultMailbox	Default location associated with user. Only returned if *Status* is **Success**.
DefaultRecipient	Default recipient associated with user. Recipient location to be used by client when an explicit recipient is not available. Only returned if *Status* is **Success**.
UserRights	An ASCII representation of a bit string describing the global rights of the user. This is a string of 64 zeros (0) and/or ones (1). Each position in the string represents a specific right.
ServerTime	Current Server local timestamp in the form – YYYYMMDDHHMMSSMMM.
ServerTimeGMT	Current Server GMT timestamp in the form – YYYYMMDDHHMMSSMMM.
ErrorId	Machine friendly representation of failure. Optional. Only exists if *Status* is **Failure**.
ErrorText	Human friendly description of failure. Optional.
IdleLifetime	Number of minutes that a session may remain idle before the server will terminate the session. This may be used by perimeter servers to timeout client sessions rather than waiting for a client action to fail due to terminated session.

Logoff

This service terminates the logical session created using the *Logon* service. If the session is active, then the session is terminate and the elapsed *SessionTime* is returned. Otherwise an error is reported.

LogoffReq

The following table explains the parameters for this service.

Parameter	Description
User	Name of Identify to be authenticated.
SessionId	Session identifier.

LogoffRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Immediate status of the message: **Success** or **Fail**.
ServerTime	Current Server local timestamp in the form – YYYYMMDDHHMMSSMMM.
ServerTimeGMT	Current Server GMT timestamp in the form – YYYYMMDDHHMMSSMMM.
ErrorId	Machine friendly representation of failure. Optional. Only exists if *Status* is **Failure**.
ErrorText	Human friendly description of failure. Optional.

Version

This service provides version information of the server.

VersionReq

There are no parameters for the version request.

VersionRsp

The following table explains the parameters for this service.

Parameter	Description
InterfaceVersion	Interface version of the server. Set to **6** for the current interface.
Version	MessageWay product version, for example, **6.1.0**.
Company	Ipswitch, Inc.
Copyright	Copyright notice in mwsi. This is also what appears when you run ./mwsi –version For instance, 'Copyright (c) 2002-2012 Ipswitch, Inc.'
FileVersion	Version of mwsi, for example, **6.1.0.1**.
Revision	Low level revision information. Same as appears on last line when you run ./mwsi –version For instance, '$LastChangedRevision: 6272 $'
ServerTime	Current Server local timestamp in the form – YYYYMMDDHHMMSSMMM.
ServerTimeGMT	Current Server GMT timestamp in the form – YYYYMMDDHHMMSSMMM.

SendMessage

This service submits a single message to MessageWay.

SendMessageReq

The following table explains the parameters for this service.

Parameter	Description
Sender	Name of message sender. May, optionally, be a MessageWay location. May be client specific. Defaults to users *DefaultMailbox*.
Recipient	Name of MessageWay destination. May contain pipelined routing information, e.g., **translate:zip:mailto:mwaysupport@ipswitch.com**. Defaults to users *DefaultRecipient*.
InputName	Client specific unique reference value. Optional. Set to *Filename* if missing or blank.
Filename	Message file name. The service interface will remove any path, remove consecutive periods and replace invalid characters with underscore (_). Optional. If missing or blank, created from *InputName*, or set to **M%msgid%.dat** if *InputName* is blank.
ClassId	Arbitrary classification of message. This is a character string up to 32 characters. Leading and trailing spaces will be removed.
ContentType	MIME type of message.
Priority	Optional. Number from 1 to 5 where the default is 3, which is the standard MessageWay message priority.
Text	If **True**, uploaded message is converted to use native line endings: CRLF on Windows, NL on other platforms. If false or missing, uploaded message is not altered.
Size	The size in bytes of *MessageContent*. Optional

Parameter	Description
Restartable	Indicates whether the message transfer should be restartable. Valid values: ▪ **T** = restartable ▪ **F** or *null* = not restartable If the transfer is restartable then the message content must be encoded in FTP block transmission mode (see RFC 959 3.4.2). Server restart markers will be returned in asynchronous HTTP responses with a response code of 199 and the following custom HTTP headers: ▪ **X-SrcMarker:** <*client-restart-marker*> ▪ **X-DestMarker:** <*server-restart-marker*> It is the responsibility of the client to save these restart markers and request the transfer be restarted, using the *RestartMarker* parameter below.
RestartMarker	Server specific restart marker, indicating the position to restart a previous failed transfer. The presence of this parameter, indicates that the client wishes to restart a previous transfer. In this case, *MessageContent* will only include the trailing part of the message starting at the corresponding client restart marker. If the previous partial message cannot be found or restart is not possible, then an error message will be returned.
MessageContent	Content of the message. The nature of this field depends on the specific interface used. For the Java interface, the message content will be in a disk file which will be referenced in the API by name. For the C++ interface the message content may be a disk file, a std:string or an object of type TMWayStream.

SendMessageRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Immediate status of the message: **Success** or **Fail**.
MessageId	Unique identifier assigned by MessageWay for this message. May be used to check status later.
Size	Received size of message. Should match *Size* in request.

Parameter	Description
ErrorId	Machine friendly representation of failure. Optional. Only exists if *Status* is **Failure**.
ErrorText	Human friendly description of failure. Optional.

GetMessageStatus

This service obtains status information of a specific message and optionally that message's output, report and/or acknowledgments. A recursive option allows the receipt of outputs of outputs in the case of pipelined processing. For instance if a message is sent to **translate:zip:***destination*, then the recursive result will include the input message, the output message(s) from MWTranslator and the final output messages from the compression service. The message may be selected either with *MessageId* or with *LocationName* and zero or more of *Filename, ClassId* and/or *Status.* If more that one message match the criteria, then the oldest message (lowest *MessageId* value), is used.

GetMessageStatusReq

The following table explains the parameters for this service.

Parameter	Description
MessageId	MessageWay unique identifier for the message. Only one of *MessageId* or *LocationName* should be present.
LocationName	MessageWay location. If present, select the oldest, lowest MessageId value message within the location. *Filename* may be used to further restrict which message to select. Only one of *MessageId* or *LocationName* should be present.
Filename	Only applicable if *LocationName* is provided. Restricts messages to pick from to those which match *Filename*.
ClassId	Only applicable if *LocationName* is provided. Restricts messages to pick from to those which match *ClassId*. May be used with *Filename* and/or *Status*.
Status	Only applicable if *LocationName* is provided. Restricts messages to pick from the those which match Status. May be used with *Filename* and/or *ClassId*.

Parameter	Description
Fields	Comma separated list of Message fields to return in response. *MessageId* is always returned. If *Fields* is *****, then all fields are returned. If missing, then the following default fields are returned: ▪ **MessageType** ▪ **Size** ▪ **Status** ▪ **Sender** ▪ **Recipient** ▪ **Filename**
Options	Types of related messages to return status for. This field may contain one or more of the following: ▪ **I** – original message ▪ **O** – outputs ▪ **R** – reports ▪ **A** – acknowledgments
Recursive	Boolean value. If *true* ,indicates that output messages should be retrieved recursively through multiple processing steps if any.

GetMessageStatusRsp

The following tables explain the parameters for this service.

Parameter	Description
Count	Total number of rows available.
ErrorId	Error number in case of failure. Optional.
ErrorText	Error text in case of failure. Optional.

The following parameters may repeat depending on request options

Parameter	Description
MessageId	MessageWay unique identifier for the message.
InputMessageId	For outputs, reports, acknowledgments this is the ID of the immediate input message. For the original input message this is the same as MessageId.

Parameter	Description
OriginalMessageId	For aliases (distribution list outputs, rules processing outputs, resent messages, etc.) this is the ID of original message. Otherwise, this is the same as MessageId.
Level	Number of steps away for original message: **0** – input message, **1** – immediate outputs, **2** – outputs of outputs, etc.
LocationName	Name of location where message resides.
MessageType	Type of message. One of the following: ▪ **Input** ▪ **Output** ▪ **Report** ▪ **Acknowledgment**
Size	Size of message in bytes.
CRLFSize	▪ Transfer size of message assuming text file with CRLF line endings.
Status	Status of message. One of the following: ▪ **Queued** ▪ **Hold** ▪ **Waiting** ▪ **Processing** ▪ **Sending** ▪ **Available** ▪ **Error** ▪ **Canceled** ▪ **Completed**
Sender	Name of message sender.
Recipient	Name of message recipient.
TimeStamp	Date and time of last message action.
IBTimeStarted	Timestamp when message upload/receipt started.
IBTimeComplete	Timestamp when message upload/receipt was complete.
OBTimeReady	Timestamp when message was ready for autodelivery or pickup.
OBTimeStarted	Timestamp when message download/autodelivery was started.
OBTimeComplete	Timestamp when message download/autodelivery was complete.
InputName	External identifier provided when message is received into MessageWay. This could be a filename or a unique ID assigned by an external messaging system.

Parameter	Description
Filename	Filename associated with the message.
OutputName	External identifier provided when message is delivered or downloaded. This could be a filename or a unique ID assigned by an external messaging system.
ClassId	Arbitrary classification of message.
ContentType	MIME type of message.
ProcessingStatus	Status from service (MWTranslator) for completed messages. One of the following: ▪ **Accepted** ▪ **Rejected** ▪ **Partially Accepted** ▪ **Accepted with Errors** ▪ **Acknowledged**, ▪ **Security Reject**

GetReconStatus

This service obtains document reconciliation status for all documents of a specific outbound message. If the message is not an output from the translator, or if logging is not enabled, then the response will contain a document count of zero.

GetReconStatusReq

The following table explains the parameter for this service.

Parameter	Description
MessageId	MessageWay unique identifier for the outbound message to return document reconciliation information.

GetReconStatusRsp

The following tables explain the parameters for this service.

Parameter	Description
Count	Total number of rows available.
ErrorId	Error number in case of failure. Optional.
ErrorText	Error text in case of failure. Optional.

The following parameters may repeat for number of documents in subject message

Parameter	Description
id	Unique ID of document within the message. Consists of comma-separated list of integers, beginning with MessageId, that represent the relative position of the wrapper levels and documents within the message. For instance, a single X12 document would have the value **20121105120000000,1,1,0,0,1**. The second document within the same functional group would have the value **20121105120000000,1,1,0,0,2**. The first document within the second functional group – **20121105120000000,1,2,0,0,1**.
MessageId	MessageWay unique identifier for the message.
TimeStamp	Date time that subject message was sent.
ReconciliationStatus	Reconciliation Status of the document. Values are: ▪ **1** – Accepted ▪ **2** – Accepted with Errors ▪ **3** – Partially Accepted ▪ **4** – Rejected ▪ **5** – Awaiting Ack ▪ **6** – Ack not Expected ▪ **7** – Acknowledged ▪ **8** – Received ▪ **9** – ICH Wrapper Accepted ▪ **10** – ICH Wrapper Accepted with Errors ▪ **11** – ICH Wrapper Rejected ▪ **12** – Security Reject ▪ **13** – Reconciliation Error ▪ **14** – Invalid Status

Parameter	Description
DocId	Document ID of subject document.
DocControlRef	Control reference of subject document.
UserField1	User defined field. Typically used to provide document specific key, such as PO Number.
UserField2 UserField3 UserField4	Additional user-defined fields.
FgControlRef	Control reference of subject functional group.
FgSender	Name of functional group message sender.
FgRecipient	Name of functional group message recipient.
IchControlRef	Control reference of subject interchange.
IchSender	Name of interchange message sender.
IchRecipient	Name of interchange message recipient.

GetMessageList

This service obtains a list of messages in a location. It may be used prior to receiving individual messages. By default this service provides a list of messages that are available for retrieval (download). Optionally a list of canceled, completed or uploaded messages may be requested. If all uploaded messages are requested then all messages with the logged on users default location as the message sender will be returned. If page length, *PageLen*, is provided, then the result will be limited to *PageLen* rows. If *PageNum* is provided, then the first *(PageNum-1) * PageLen* rows will be skipped.

GetMessageListReq

The following table explains the parameters for this service.

Parameter	Description
LocationName	Name of MessageWay location from which to retrieve messages. The authenticated user must have download access to this location. If *Status* is **Uploaded**, then *LocationName* may be blank in which case all messages on the system with the logged on users default location, as the message sender will be returned.
Filename	If provided, only messages with this *Filename* are retrieved. The asterisk (*) may be used as a wildcard.
ClassId	If provided, only messages with this *ClassId* are retrieved.
Status	Status of Message. One of the following: ▪ **Available** (default) ▪ **Canceled** ▪ **Completed** ▪ **Uploaded** ▪ **Downloaded** ▪ **AvailableFS** (one of **A**, **Q**, **H**, **W**) ▪ **BusyFS** (one of **S**, **R**, **U**, **D**, **P**, **O**, **X**, **Z**)
PageNum	If present and *PageLen* is also present , skips the first ((*PageNum*-1) * *PageLen*) rows. If missing, the first page is returned.
PageLen	Maximum number of lines to return in response. If zero, or missing then *PageNum* will be ignored and all lines will be returned.
Fields	Comma-separated list of Message fields to return in response. *MessageId* is always returned. If *Fields* is *****, then all fields are returned. If missing, then the following default fields are returned: ▪ **MessageType** ▪ **Size** ▪ **Status** ▪ **Sender** ▪ **Recipient** ▪ **Filename**

GetMessageListRsp

The following tables explain the parameters for this service.

Parameter	Description
Count	Total number of rows available.
RowCount	Sequence of first returned row in response.
PageCount	Total number of available pages.
ServerTime	Current Server local timestamp in the form – YYYYMMDDHHMMSSMMM.
ServerTimeGMT	Current Server GMT timestamp in the form – YYYYMMDDHHMMSSMMM.
ErrorId	Error number in case of failure. Optional.
ErrorText	Error text in case of failure. Optional.

The following parameters may repeat:

Parameter	Description
MessageId	MessageWay unique identifier for the message.
InputMessageId	For outputs, reports, acknowledgments this is the Id of the immediate input message. For the original input message this is the same as MessageId.
OriginalMessageId	For aliases (Distribution list outputs, rules processing outputs, resent messages, etc.) this is the Id of original message. Otherwise this is the same as MessageId.
LocationName	Name of location that message resides in.
MessageType	Type of message – one of (Input, Output, Report, Acknowledgment).
Size	Size of message in bytes.
Status	Status of Message. One of (Queued, Hold, Waiting, Processing, Sending, Available, Error, Canceled, Completed)
Sender	Name of message sender.
Recipient	Name of message recipient.
TimeStamp	Date and time of last message action.
IBTimeStarted	Timestamp when message upload/receipt started.
IBTimeComplete	Timestamp when message upload/receipt was complete.
OBTimeReady	Timestamp when message was ready for autodelivery or pickup.

Parameter	Description
OBTimeStarted	Timestamp when message download/autodelivery was started.
OBTimeComplete	Timestamp when message download/autodelivery was complete.
InputName	External identifier provided when message is received into MessageWay. This could be a filename or a unique ID assigned by an external messaging system.
Filename	Filename associated with the message.
OutputName	External identifier provided when message is delivered or downloaded. This could be a filename or a unique ID assigned by an external messaging system.
ClassId	Arbitrary classification of message.
ContentType	MIME type of message.
ProcessingStatus	Status from processing gateway (MWTranslator) for completed messages. One of (Accepted, Rejected, Partially Accepted, Accepted with Errors, Acknowledged, Security Reject)

ReceiveMessage

This service receives a single message from MessageWay. Either a specific message may be received, or the first available message at a location may be received. A specific message is received by providing the *MessageId* of that message. Otherwise, the *LocationName* must be supplied and the first (lowest *MessageId* value) message matching the optional fields *Status*, *Filename* and *ClassId* will be received. *Status* defaults to **Available** if not provided.

ReceiveMessageReq

The following table explains the parameters for this service.

Parameter	Description
MessageId	MessageWay unique identifier for the message. Only one of *MessageId* or *LocationName* should be present.
LocationName	MessageWay location. If present, select the oldest (lowest *MessageId* value) message within the location. *Filename*, *ClassId* and/or *Status* may be used to further restrict which message to select. Only one of *MessageId* or *LocationName* should be present.

Parameter	Description
Filename	Restricts messages to pick from to those which match *Filename*. The asterisk (*) may be used as a wildcard. This field may also be used with *ClassId* and/or *Status*.
ClassId	Restricts messages to pick from to those which match *ClassId*. This field may be used with *Filename* and/or *Status*.
Status	If present, only messages with a matching status will be retrieved. Valid values are: ▪ **Available** ▪ **Canceled** ▪ **Completed** If *MessageId* is not present, then *Status* defaults to **Available**.
Compress	If set to true, the retrieved message will be compressed to zip file format.
View	If set to **true**, then the message will be transferred for viewing only. There will be no updates of the message meta-data, including status and timestamps.
Mode	Specifies the behavior for transferring (downloading) the file. ▪ **FileSystem** – create an alias message for the download so the original file remains available, and multiple downloads may be tracked separately. ▪ **Messaging** – legacy behavior where the file is no longer available once the transfer begins.
Eol	End-of-line characters: **CRLF** or **NL**. If present, message is assumed to have a content-type of *text* (even if it doesn't) and is retrieved with the specified line endings.
Restartable	Indicates whether the message transfer should be restarted. Valid values: ▪ **T** = restartable ▪ **F** or *null* = not restartable. If the transfer is restartable, then the message content will be returned in FTP block transmission mode (see RFC 959 3.4.2). It is the responsibility of the client to decode the block format, save the restart marker (along with a client restart marker) and request the transfer be restarted, using the *RestartMarker* parameter below.
RestartMarker	Server-specific restart marker, indicating the position to restart a previous failed transfer. The presence of this parameter indicates that the client wishes to restart a previous transfer. If the previous partial message cannot be found or restart is not possible, then an error message will be returned. Otherwise, the returned content type will resume at the point indicated by *RestartMarker*.

Parameter	Description
Fields	Comma-separated list of message fields to return in response. *MessageId* is always returned. If *Fields* is *****, then all fields are returned. If missing, then the following default fields are returned: ▪ **MessageType** ▪ **Size** ▪ **Status** ▪ **Sender** ▪ **Recipient** ▪ **Filename**
Confirm	If set to **true**, the client will confirm receipt of the message with *MessageReceivedReq*.

ReceiveMessageRsp

The following table explains the parameters for this service.

Parameter	Description
MessageId	MessageWay unique identifier for the message.
InputMessageId	For outputs, reports, acknowledgments this is the ID of the immediate input message. For the original input message, this is the same as *MessageId*.
OriginalMessageId	For aliases (distribution list outputs, rules processing outputs, resent messages, etc.) this is the ID of original message. Otherwise, this is the same as *MessageId*.
LocationName	Name of location where message resides.
MessageType	Type of message. One of the following: ▪ **Input** ▪ **Output** ▪ **Report** ▪ **Acknowledgment**
Size	Size of message in bytes.

Parameter	Description
Status	Status of Message. One of the following: ▪ **Queued** ▪ **Hold** ▪ **Waiting** ▪ **Processing** ▪ **Sending** ▪ **Available** ▪ **Error** ▪ **Canceled** ▪ **Completed**
Sender	Name of message sender.
Recipient	Name of message recipient.
TimeStamp	Date and time of last message action.
IBTimeStarted	Timestamp when message upload/receipt started.
IBTimeComplete	Timestamp when message upload/receipt was complete.
OBTimeReady	Timestamp when message was ready for autodelivery or pickup.
OBTimeStarted	Timestamp when message download/autodelivery was started.
OBTimeComplete	Timestamp when message download/autodelivery was complete.
InputName	External identifier provided when message is received into MessageWay. This could be a file name or a unique ID assigned by an external messaging system.
Filename	File name associated with the message.
OutputName	External identifier provided when message is delivered or downloaded. This could be a file name or a unique ID assigned by an external messaging system.
ClassId	Arbitrary classification of message.
ContentType	MIME type of message.
ProcessingStatus	Status from service (MWTranslator) for completed messages. One of the following: ▪ **Accepted** ▪ **Rejected** ▪ **Partially Accepted** ▪ **Accepted with Errors** ▪ **Acknowledged** ▪ **Security Reject**

Parameter	Description
MessageContent	Content of the message. The nature of this field depends on the specific interface used. For the Java interface, the message content will be in a disk file, which will be referenced in the API by name.
ErrorId	Error number in case of failure. Optional.
ErrorText	Error text in case of failure. Optional.

ReleaseMessage

This service releases a message, if that message is on hold, wait, or output hold.

ReleaseMessageReq

The following table explains the parameter for this service.

Parameter	Description
MessageId	MessageWay unique identifier for the message.

ReleaseMessageRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Result of action: **Success** or **Fail**.
ErrorId	Error number if *Status* is **Fail**.
ErrorText	Error text if *Status* is **Fail**.

CancelMessage

This service cancels a message. Only one of *MessageId* or *(LocationName,Filename,ClassId,Status)* should be present to identify the message to be canceled.

At most one message will be canceled. If more than one message matches the request parameters, then the oldest (determined by the message Id value) will be canceled.

CancelMessageReq

The following table explains the parameters for this service.

Parameter	Description
MessageId	MessageWay unique identifier for the message.
LocationName	Name of location containing message to cancel.
Filename	Filename (without path) of message to cancel.
ClassId	ClassId of message to cancel. Only applicable if *LocationName* and *Filename* are provided.
Status	Status of message to cancel. Only applicable if *LocationName* and *Filename* are provided. ▪ **Available** (default) ▪ **Completed** ▪ **Canceled**

CancelMessageRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Result of action: **Success** or **Fail**.
ErrorId	Error number if *Status* is **Fail**.
ErrorText	Error text if *Status* is **Fail**.

PullMessage

This service triggers input polling for an adapter location (site) or triggers execution of a service location.

PullMessageReq

The following table explains the parameter for this service.

Parameter	Description
LocationName	MessageWay site or service location.

PullMessageRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Result of action: **Success** or **Fail**.
ErrorId	Error number if *Status* is **Fail**.
ErrorText	Error text if *Status* is **Fail**.

CreateMessage

This service creates a new MessageWay message. It should be followed with one or more *WriteMessage* requests followed by a *CommitMessage* or *AbortMessage* request – all on the same connection. If the *CommitMessage* request has not been received and the connection is closed, then the message will be aborted.

CreateMessageReq

The following table explains the parameters for this service.

Parameter	Description
Sender	Name of message sender. May optionally be a MessageWay location. May be client specific. Defaults to user's *DefaultMailbox*.

Parameter	Description
Recipient	Name of MessageWay destination. May contain pipelined routing information, e.g. **Translate:zip:mailto:mwaysupport@ipswitch.com**. Defaults to users *DefaultRecipient*.
InputName	Client specific unique reference value. Optional. Set to *Filename* if missing or blank.
Filename	Message file name. The service interface will remove any path and replace invalid characters with underscore (_). Optional. If missing or blank, created from *InputName* or set to **M%msgid%.dat** if *InputName* is blank.
RestartMessageId	Optional. MessageId of restarted transfer. Data will be appended to the end of this message. Only applicable if Restart is *true*. **NOTE:** This supports the Append function in the MessageWay SFTP Perimeter Server.
ClassId	Arbitrary classification of message. This is a character string up to 32 characters – leading and trailing spaces will be removed.
ContentType	MIME type of message.
Text	If **true**, uploaded message is converted to use native line endings: CRLF on Windows, NL on other platforms. If **false** or missing, uploaded message is not altered.
Restart	If **true**, restart transfer from end of already transferred data. In this case transfer data will be appended to the end of the current message. **NOTE:** This supports the Append function in the MessageWay SFTP Perimeter Server.

CreateMessageRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Immediate status of the message: **Success** or **Fail**.
MessageId	Unique identifier assigned by MessageWay for this message. Must be used in subsequent *WriteMessage* / *CommitMessage* / *AbortMessage* requests.
ErrorId	Machine friendly representation of failure. Optional. Only exists if *Status* is **Fail**.
ErrorText	Human friendly description of failure. Optional.

WriteMessage

This service writes data to a new message created by a call to *CreateMessage*. Only one message may be created at a time on a single connection.

WriteMessageReq

The following table explains the parameters for this service.

Parameter	Description
MessageId	Unique identifier returned from the *CreateMessage* request.
RestartMarker	Position (zero relative offset from start of file) to start writing data. If omitted, start at end of last write.
MessageContent	Content or partial content from the message. This may be any arbitrary binary data. The nature of this field depends on the specific interface used.

WriteMessageRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Immediate status of the message: **Success** or **Fail**.
ErrorId	Machine friendly representation of failure. Optional. Only exists if *Status* is **Fail**.
ErrorText	Human friendly description of failure. Optional.

CommitMessage

This service commits a previously created MessageWay message. It must follow a *CreateMessage* request and, optionally, one or more *WriteMessage* requests.

CommitMessageReq

The following table explains the parameter for this service.

Parameter	Description
MessageId	Unique identifier returned from the *CreateMessage* request.

CommitMessageRsp

The following table explains the parameters for this service.

Parameter	Description
Size	Total size in bytes of message committed. This should equal the sum of the message content sizes (implicit) provided with *WriteMessage* requests.
Status	Immediate status of the message: **Success** or **Fail**.
ErrorId	Machine friendly representation of failure. Optional. Only exists if *Status* is **Fail**.
ErrorText	Human friendly description of failure. Optional.

OpenMessage

This service opens an existing MessageWay message for subsequent read. It should be followed with one or more *ReadMessage* requests followed by a *CloseMessage* or *AbortMessage* request – all on the same connection. If the *CloseMessage* request has not been received and the connection is closed, then the message will be aborted.

If *View* is not set to **true**, the status of the message will be changed to **Downloading** upon successful completion of this service. The previous status will be saved in case the message is aborted.

Either a specific message may be opened, or the first available message at a location may be opened. A specific message is opened by providing the *MessageId* of that message. Otherwise the *LocationName*

must be supplied and the first (lowest *MessageId* value) message matching the optional fields *Status*, *Filename* and *ClassId* will be opened. *Status* defaults to **Available** if not provided. *OpenMessage*, *ReadMessage*, and *CloseMessage* together provide the same function as *ReceiveMessage*.

OpenMessageReq

The following table explains the parameters for this service.

Parameter	Description
MessageId	MessageWay unique identifier for the message. Only one of *MessageId* or *LocationName* should be present.
LocationName	MessageWay location. If present, select the oldest (lowest MessageId value) message within the location. *Filename*, *ClassId* and/or *Status* may be used to further restrict which message to select. Only one of *MessageId* or *LocationName* should be present.
Filename	Only applicable if *LocationName* is provided. Restricts messages to pick from to those which match *Filename*. May also be used with *ClassId* and/or *Status*.
ClassId	Only applicable if *LocationName* is provided. Restricts messages to pick from to those which match *ClassId*. May be used with *Filename* and/or *Status*.
Status	If present, only messages with a matching status will be opened. Use with *LocationName* to open **Canceled** or **Completed** messages. **Available** messages are the default.
OutputName	External identifier provided when message is delivered or downloaded. This could be a filename or a unique ID assigned by an external messaging system.
Mode	Specifies the behavior for transferring (downloading) the file. ▪ **FileSystem** – create an alias message for the download so the original file remains available, and multiple downloads may be tracked separately. ▪ **Messaging** – legacy behavior where the file is no longer available once the transfer begins.
Compress	If set to **true**, the retrieved message will be compressed to zip file format.
Eol	End-of-line characters: **CRLF** or **NL**. If present, message is assumed to have a content-type of *text* (even if it doesn't) and is retrieved with the specified line endings.
View	If set to **true**, then the message will be opened for viewing only - there will be no updates of the message meta-data including status and timestamps.
Fields	Comma-separated list of Message fields to return in response. If missing, then all non-optional fields will be returned, for example: **MessageId,Size,TimeStamp,Sender**.

OpenMessageRsp

The following table explains the parameters for this service.

Parameter	Description
MessageId	MessageWay unique identifier for the message.
MessageType	Type of message. One of the following: ▪ **Input** ▪ **Output** ▪ **Report** ▪ **Acknowledgment**
InputName	External identifier provided when message is received into MessageWay. This could be a file name or a unique ID assigned by an external messaging system.
Filename	*Filename* attribute of the message. This will be a valid file name without a directory.
OutputName	External identifier provided when message is delivered or downloaded. This could be a file name or a unique ID assigned by an external messaging system.
ClassId	Arbitrary classification of message.
ContentType	MIME type of message.
Size	Size of message in bytes.
Status	Status of Message. One of the following: ▪ **Queued** ▪ **Hold** ▪ **Waiting** ▪ **Processing** ▪ **Sending** ▪ **Available** ▪ **Error** ▪ **Canceled** ▪ **Completed**
TimeStamp	Date and time of last message action.
IBTimeStarted	Date and time of start of message receipt. Optional.
Sender	Name of message sender.

Parameter	Description
Recipient	Name of message recipient.
ErrorId	Error number in case of failure. Optional.
ErrorText	Error text in case of failure. Optional.

ReadMessage

This service reads data from an existing message previously selected with an *OpenMessage* request.

ReadMessageReq

The following table explains the parameters for this service.

Parameter	Description
MessageId	Unique identifier returned from the *CreateMessage* request.
Size	Maximum size of *MessageContent* to be returned.

ReadMessageRsp

The following table explains the parameters for this service.

Parameter	Description
MessageContent	Content or partial content from the message. This may be any arbitrary binary data. The nature of this field depends on the specific interface used.
Status	Immediate status of the message: **Success**, **EOM**, or **Fail**.
ErrorId	Machine friendly representation of failure. Optional. Only exists if *Status* is **Fail**.
ErrorText	Human friendly description of failure. Optional.

CloseMessage

This service closes a previously opened MessageWay message. It must follow an *OpenMessage* request and optionally one or more *ReadMessage* requests. The status of the message will be changed to **Completed**. To avoid this, invoke *AbortMessage* instead of *CloseMessage*.

CloseMessageReq

The following table explains the parameters for this service.

Parameter	Description
MessageId	Unique identifier returned from the *OpenMessage* request.
OutputName	Client-specific, unique reference value. Optional. If present, then the *OutputName* of the message will be set to this value.

CloseMessageRsp

The following table explains the parameters for this service.

Parameter	Description
MessageId	Unique identifier returned from the *CloseMessage* request.
Status	Immediate status of the message: **Success** or **Fail**.
ErrorId	Machine friendly representation of failure. Optional. Only exists if *Status* is **Fail**.
ErrorText	Human friendly description of failure. Optional.

AbortMessage

This service aborts a previously created or opened MessageWay message. It must follow a *CreateMessage* or *OpenMessage* request. If the message was opened, the the status will be changed back to the status before the *MessageOpen* request. If the message was created, then the newly created message will be discarded.

AbortMessageReq

The following table explains the parameter for this service.

Parameter	Description
MessageId	Unique identifier returned from the *OpenMessage* request.

AbortMessageRsp

The following table explains the parameters for this service.

Parameter	Description
MessageId	Unique identifier returned from the *AbortMessage* request.
Status	Immediate status of the message: **Success** or **Fail**.
ErrorId	Machine friendly representation of failure. Optional. Only exists if *Status* is **Fail**.
ErrorText	Human friendly description of failure. Optional.

GetLocationRights

This service returns the effective rights that the logged on user has for a specific location.

GetLocationRightsReq

The following table explains the parameter for this service.

Parameter	Description
LocationName	Name of site, service location or pickup mailbox.
IsRecipient	If set to **true**, then LocationName will be treated as a recipient, which may consist of multiple locations: either comma-separated for dynamic distribution lists and/or colon-separated for compound recipients. In this case, the returned *Rights* will be calculated as only those rights allowed by *all* included locations.

GetLocationRightsRsp

The following table explains the parameters for this service.

Parameter	Description
LocationName	Name of site, service location or pickup mailbox.
Rights	An ASCII representation of a bit string describing the effective rights of the logged user for *LocationName*. This is a string of 64 zeros (0) and/or ones (1). Each position in the string represents a specific right. Optional. Only provided if the user has *some* rights to *LocationName*. Otherwise, *ErrorId* and *ErrorText* will be provided.
ErrorId	Machine friendly representation of failure. Optional.
ErrorText	Human friendly description of failure. Optional.

ResendMessage

This service re-queues a message that is complete or in error for auto-delivery.

ResendMessageReq

The following table explains the parameter for this service.

Parameter	Description
MessageId	Message Id of the message to be resent.

ResendMessageRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Status of action: **Success** or **Fail**.
ErrorId	Machine friendly representation of failure. Optional.
ErrorText	Human friendly description of failure. Optional.

RedirectMessage

This service re-directs a message that is complete or in error to a new recipient. It can also be used to change the filename property.

RedirectMessageReq

The following table explains the parameters for this service.

Parameter	Description
MessageId	Message ID of the message to be redirected.
Recipient	New recipient.
Filename	New filename. Optional.

RedirectMessageRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Status of action: **Success** or **Fail**.
ErrorId	Machine friendly representation of failure. Optional.
ErrorText	Human friendly description of failure. Optional.

MessageReceived

This service confirms that a message received via *ReceiveMessageReq* was successfully received by the client.

MessageReceivedReq

The following table explains the parameters for this service.

Parameter	Description
MessageId	Message Id of the received message (from *ReceiveMessageRsp*).
Status	Status of receipt: **Success** or **Fail**. If success, the message is marked complete; otherwise the message returns to the status prior to **Downloading**. This is usually **Available** or **Error**.
OutputName	External name given to message by client. Optional.
ErrorText	Human friendly description of failure. Optional.

MessageReceivedRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Status of action.: **Success** or **Fail**.
ErrorId	Machine friendly representation of failure. Optional.
ErrorText	Human friendly description of failure. Optional.

GetNonPrivGlobalFlags

This service returns global flags defined in MessageWay. Currently returns only the FIPSOnlyTransport flag.

GetNonPrivGlobalFlagsReq

N/A

GetNonPrivGlobalFlagsRsp

The following table explains the parameters for this service.

Parameter	Description
FIPSOnlyTransport	**T** or **F**
Status	Status of action: **Success**

GetLocationList

This service obtains a list of child locations for a location. By default this service returns a list of all child locations of the parent. Optionally a list of locations by *Type* and or *State* may be requested. If *PageLen* is provided, then the result will be limited to *PageLen* rows.If *PageNum* is provided, then the first *(PageNum-1) * PageLen* rows will be skipped.

GetLocationListReq

The following table explains the parameters for this service.

Parameter	Description
LocationName	Required. Name of MessageWay parent location to retrieve child locations of. The authenticated user must have access to this location.

Parameter	Description
Type	Optional. If provided, only locations of this Type are retrieved. One of the following: ▪ **All** (default) ▪ **Processing** ▪ **Input** ▪ **Output** ▪ **IO** ▪ **Generic** ▪ **System** ▪ **Pickup**
State	Optional: State of Location. One of the following: ▪ **All** (default) ▪ **Active** ▪ **Hold** ▪ **Exception**
PageNum	Optional. If present and *PageLen* is also present , skips the first *((PageNum-1)* * *PageLen)* rows. If missing, the first page is returned.
PageLen	Optional: Maximum number of lines to return in response. If zero, or missing then *PageNum* will be ignored and all lines will be returned.
Fields	Comma-separated list of Location fields to return in response. *LocationName* is always returned. If *Fields* is *****, then all fields are returned. If missing, then the following default fields are returned: *LocationName*, *Type*, *State*, *OutputState*, *SchedState*, *UserRights*, *LastModified*.

GetLocationListRsp

The following table explains the parameters for this service.

Parameter	Description
Location	Request Parent Location.
Type	Request Type. Default is **All**
State	Request State. Default is **All**
Count	Total number of rows available.

Parameter	Description
PageNum	Page number of returned page.
PageCount	Total number of available pages.
ServerTime	Current Server local timestamp in the form – YYYYMMDDHHMMSSMMM.
ServerTimeGMT	Current Server GMT timestamp in the form – YYYYMMDDHHMMSSMMM.
ErrorId	Error number in case of failure. Optional.
ErrorText	Error text in case of failure. Optional.

The following parameters may repeat:

Parameter	Description
LocationName	Name of the child location.
Type	Type of location. One of: ▪ **Processing** ▪ **Input** ▪ **Output** ▪ **Both** ▪ **Generic** ▪ **System** ▪ **Pickup**
State	State of location. One of: ▪ **Active** ▪ **Hold** ▪ **Exception**
OutputState	Output State of location (A – active, H - hold).
SchedState	Schedule State of location (O – open, C - closed).
UserRights	An ASCII representation of a bit string describing the effective rights of the logged user for LocationName. This is a string of 64 '0's and/or '1's. Each position in the string represents a specific right.
LastModified	Last Modified Timestamp in the form – YYYYMMDDHHMMSSMMM.

CreateLocation

This service creates a location in the File System folder.

CreateLocationReq

The following table explains the parameters for this service.

Parameter	Description
LocationName	Pickup mailbox location name.

CreateLocationRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Result of action. One of: ▪ **Success** ▪ **Fail**
ErrorId	Error number in case of failure. Optional.
ErrorText	Error text in case of failure. Optional.

DeleteLocation

This service deletes a location in the File System folder.

NOTE: The delete operation will fail if the location is not in the File System folder, if the location is the root location, or if the location has messages in it.

DeleteLocationReq

The following table explains the parameters for this service.

Parameter	Description
LocationName	Pickup mailbox location name.

DeleteLocationRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Result of action. One of: ▪ **Success** ▪ **Fail**
ErrorId	Error number in case of failure. Optional.
ErrorText	Error text in case of failure. Optional.

RenameLocation

This service renames a location in the File System folder.

NOTE: Rename will fail if the location is the root location or the default location of any user.

RenameLocationReq

The following table explains the parameters for this service.

Parameter	Description
LocationName	Pickup mailbox location name.
NewLocationName	New pickup mailbox location name.

RenameLocationRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Result of action. One of: ▪ **Success** ▪ **Fail**
ErrorId	Error number in case of failure. Optional.
ErrorText	Error text in case of failure. Optional.

MarkMessage

This service marks a message for append.

MarkMessageReq

The following table explains the parameters for this service.

Parameter	Description
MessageId	Message ID of the message to be marked for append.
Status	The status of the message to be marked for append. Status 'FS' (filesystem) one of: ▪ **A** ▪ **Q** ▪ **H** ▪ **W**

MarkMessageRsp

The following table explains the parameters for this service.

Parameter	Description
Status	Result of action. One of: ▪ **Success** ▪ **Fail**

Parameter	Description
ErrorId	Error number in case of failure. Optional.
ErrorText	Error text in case of failure. Optional.

Field Values for Rights Field

The bit positions of the Rights field are defined as follows, starting with the left-most position: For example, 00000010000001011100000000000000000000000000000000000000000000000000 gives *Read Properties*, *Read Message Properties*, *View Messages*, *Upload Messages* and *Download Messages* rights.

Bit Position	Right
0	Administer Users
1	Modify Access Rights
2	Modify Global Properties
3	Read Adapter/Service Properties
4	Modify Adapter/Service Properties
5	Start Stop Adapter/Service
6	Read Properties
7	Modify Properties
8	Rename
9	Delete
10	Create
11	Perform Location Actions
12	Perform Message Actions
13	Read Message Properties
14	Modify Message Properties
15	View Messages
16	Upload Messages
17	Download Messages

Bit Position	Right
18	View System Counts
19	View Adapters/Services
20	Maker
21	Checker
22	Create Reports - ONEview
23	Change Reports - ONEview
24	Generate Reports - ONEview
25	Schedule Reports - ONEview
26	View Reports - ONEview
27	Create Dashboard - ONEview
28	View Dashboard - ONEview
29	Change Dashboard - ONEview
30	Add Report to Dashboard - ONEview
31	Super User
32	Change ONEview UI - ONEview
33	Manage Organizations
34	Manage Realms
35-63	not used

Error Codes

The following errors codes may be returned in *ErrorId* fields:

ErrorId	ErrorText	Notes
101	Unsupported Request	The XML request document is not supported in this context. A request other the *SendMessageReq* was sent with attached message content.
102	Invalid Request	The XML request document is not valid.
103	Invalid User Id/Password	MessageWay authentication failed.
104	The Password has expired	The password must be changed.
105	Access Denied	The logged on user does not have rights to the location/message accessed.
106	A parameter value is invalid	
107	A mandatory parameter is missing	
108	The request failed.	
109	A database query failed	
110	A received file is empty	
111	Unable to obtain message Id from server.	
112	There are no available messages in location.	

HTTP Interface

The MessageWay Service Interface will be implemented as XML messages passed over HTTP. Other interfaces (Java API, Web Services Interface) will use the HTTP Interface on the back-end to provide MessageWay services.

All services will be implemented with an HTTP POST / Response interaction. In most cases the body of the HTTP message will be a single XML document. The DTDs for these request/response documents are provided at the end of this document.

Custom HTTP headers, *X-MWay-User* and *X-MWay-Password*, may be used to authenticate the request, in which case the optional *User* and *Password* attributes are not required on the XML request.

X-MWay-User: *username*

X-MWay-Password: *password-hash*

The *SendMessage* and *ReceiveMessage* services require the transfer of message content. This content may be any arbitrary binary content, although typically it would be EDI or application data. This message content is separate from the XML request and response messages (*SendMessageReq*, *ReceiveMessageRsp*) that it is associated with. In these two instances only, the XML document and the message content will be transferred as two distinct parts using MIME multipart encoding.

```
...
Content-type: multipart/mixed; boundary="==================jUDfs&8$3JkSe"
...


--==================jUDfs&8$3JkSe
Content-Length: nnn


<?xml version="1.0" />
<SendMessageReq>
...
</SendMessageReq>


--==================jUDfs&8$3JkSe
```

```
Content-Length: nnn

{data…}

--==================jUDfs&8$3JkSe
```

DTD

```
<!ELEMENT Sender (#PCDATA)>

<!ELEMENT Recipient (#PCDATA)>

<!ELEMENT InputName (#PCDATA)>

<!ELEMENT Filename (#PCDATA)>

<!ELEMENT OutputName (#PCDATA)>

<!ELEMENT ClassId (#PCDATA)>

<!ELEMENT ContentType (#PCDATA)>

<!ELEMENT Priority (#PCDATA)>

<!ELEMENT Size (#PCDATA)>

<!ELEMENT MessageId (#PCDATA)>

<!ELEMENT InputMessageId (#PCDATA)>

<!ELEMENT MessageType (#PCDATA)>

<!ELEMENT Status (#PCDATA)>

<!ELEMENT TimeStamp (#PCDATA)>

<!ELEMENT ProcessingStatus (#PCDATA)>

<!ELEMENT ReconciliationStatus (#PCDATA)>

<!ELEMENT DocId (#PCDATA)>

<!ELEMENT DocControlRef (#PCDATA)>

<!ELEMENT FgControlRef (#PCDATA)>

<!ELEMENT IchControlRef (#PCDATA)>

<!ELEMENT FgSender (#PCDATA)>

<!ELEMENT FgRecipient (#PCDATA)>

<!ELEMENT IchSender (#PCDATA)>

<!ELEMENT IchRecipient (#PCDATA)>

<!ELEMENT Options (#PCDATA)>

<!ELEMENT Recursive (#PCDATA)>
```

```
<!ELEMENT LocationName (#PCDATA)>
<!ELEMENT UserField1 (#PCDATA)>
<!ELEMENT UserField2 (#PCDATA)>
<!ELEMENT UserField3 (#PCDATA)>
<!ELEMENT UserField4 (#PCDATA)>
<!ELEMENT Retry (#PCDATA)>
<!ELEMENT MessageContent (#PCDATA)>  <!-- Shall be base-64 encoded -->
<!ELEMENT UserRights (#PCDATA)>
<!ELEMENT PageNum (#PCDATA)>
<!ELEMENT PageLen (#PCDATA)>
<!ELEMENT PageCount (#PCDATA)>
<!ELEMENT Compress (#PCDATA)>
<!ELEMENT DefaultMailbox (#PCDATA)>
<!ELEMENT Type (#PCDATA)>
<!ELEMENT State (#PCDATA)>
<!ELEMENT OutputState (#PCDATA)>
<!ELEMENT SchedState (#PCDATA)>
<!ELEMENT ServerTime (#PCDATA)>
<!ELEMENT ServerTimeGMT (#PCDATA)>
<!ELEMENT InterfaceVersion (#PCDATA)>
<!ELEMENT Company (#PCDATA)>
<!ELEMENT Copyright (#PCDATA)>
<!ELEMENT Version (#PCDATA)>
<!ELEMENT FileVersion (#PCDATA)>
<!ELEMENT Revision (#PCDATA)>
<!ELEMENT PerimeterType (#PCDATA)>
<!ELEMENT PerimeterName (#PCDATA)>
<!ELEMENT SourceIp (#PCDATA)>
<!ELEMENT AutoLogoff (#PCDATA)>
```

```
<!ELEMENT IdleLifetime (#PCDATA)>


<!ELEMENT Error (#PCDATA)>

<!ATTLIST Error

          ErrorId (CDATA) #REQUIRED>

<!ELEMENT Message (InputMessageId?, OriginalMessageId?, LocationName?, MessageType?, Size?, Status?,
Sender?, Recipient?, TimeStamp?, IBTimeStarted?, IBTimeComplete?, OBTimeReady?, OBTimeStarted?,
OBTimeComplete?, InputName?, Filename?, OutputName?, ClassId?, ContentType?, ProcessingStatus?,
Level?)>

<!ATTLIST Message

          MessageId CDATA #REQUIRED>

<!ELEMENT Messages (Message*)>

<!ATTLIST Messages

          Count CDATA #REQUIRED>

<!ELEMENT FIPSOnlyTransport (#PCDATA)>

<!ELEMENT DocumentStatus (MessageId, TimeStamp, ReconciliationStatus, DocId, DocControlRef?,
UserField1?, UserField2?, UserField3?, UserField4?, FgControlRef?, FgSender?, FgRecipient?,
IchControlRef?, IchSender?, IchRecipient?)>

<!ATTLIST DocumentStatus

          id CDATA #REQUIRED>

<!ELEMENT Documents (DocumentStatus*)>

<!ATTLIST Documents

          Count CDATA #REQUIRED>

<!ELEMENT Location (Type?, State?, OutputState?, SchedState?, UserRights?, TimeStamp?)>

<!ATTLIST Location

          LocationName CDATA #REQUIRED>

<!ELEMENT Locations (Location*)>

<!ATTLIST Locations

          Count CDATA #REQUIRED>
```

Logon

```
<!ELEMENT LogonReq (PerimeterType?, PerimeterName?, SourceIp?, AutoLogoff?)>

<!ATTLIST LogonReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>


<!ELEMENT LogonRsp (Status, SessionId, UserRights, DefaultMailbox, DefaultRecipient, ServerTime,
ServerTimeGMT, Error?)>

<!ATTLIST LogonRsp

     Version CDATA #REQUIRED

     Challenge CDATA #IMPLIED>
```

Logoff

```
<!ELEMENT LogoffReq>

<!ATTLIST LogoffReq

     User CDATA #REQUIRED

     SessionId CDATA #REQUIRED>


<!ELEMENT LogoffRsp (Status, ServerTime, ServerTimeGMT, Error?)>
```

Version

<!ELEMENT VersionReq>

<!ELEMENT VersionRsp (InterfaceVersion, Version, Company, Copyright, FileVersion, Revision, ServerTime, ServerTimeGMT)>

SendMessage

```
<!ELEMENT SendMessageReq (Sender?, Recipient, InputName?, Filename?, ClassId?, ContentType?, Size?,
Restartable?, RestartMarker?, DigestAlgorithm?)>

<!ATTLIST SendMessageReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!-- MessageContent is transferred separately -->

<!ELEMENT SendMessageRsp (Status, MessageId?, Size?, Error?)>
```

GetMessageStatus

```
<!ELEMENT GetMessageStatusReq ((MessageId | (LocationName, Filename?)), Options?, Recursive?)>

<!ATTLIST GetMessageStatusReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED
```

```
     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!ELEMENT GetMessageStatusRsp (Messages, Error?)>
```

GetReconStatus

```
<!ELEMENT GetReconStatusReq (MessageId)>

<!ATTLIST GetReconStatusReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!ELEMENT GetReconStatusRsp (Documents, Error?)>
```

GetMessageList

```
<!ELEMENT GetMessageListReq (LocationName, Filename?, ClassId?, Status?, PageNum?, PageLen?,
Fields?)>

<!ATTLIST GetMessageListReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!ELEMENT GetMessageListRsp (PageNum, PageCount, Messages, ServerTime, ServerTimeGMT, Error?)>
```

ReceiveMessage

```
<!ELEMENT ReceiveMessageReq (MessageId | (LocationName, Filename?, ClassId?, Status?)), Compress?,
View?, Mode?, Eol?, Restartable?, RestartMarker?, Fields?, Confirm?,

RequestDigestAlgorithm?, RequestSignature?)>

<!ATTLIST ReceiveMessageReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!ELEMENT ReceiveMessageRsp (Message, Error?)>

<!-- MessageContent is transferred separately →
```

ReleaseMessage

```
<!ELEMENT ReleaseMessageReq (MessageId)>

<!ATTLIST ReleaseMessageReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!ELEMENT ReleaseMessageRsp (Status, Error?)>
```

CancelMessage

```
<!ELEMENT CancelMessageReq (MessageId | (LocationName, Filename, ClassId?, Status?))>

<!ATTLIST CancelMessageReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!ELEMENT CancelMessageRsp (Status, Error?)>
```

PullMessage

```
<!ELEMENT PullMessageReq (LocationName)>

<!ATTLIST PullMessageReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!ELEMENT PullMessageRsp (Status, Error?)>
```

CreateMessage

```
<!ELEMENT CreateMessageReq (Sender?, Recipient?, ClassId?, ContentType?, InputName?, Filename?, Text?, Restart?,
RestartMessageId?)>

<!ATTLIST CreateMessageReq
```

```
     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED>

<!ELEMENT CreateMessageRsp (Status, MessageId?, Error?)>
```

WriteMessage

```
<!ELEMENT WriteMessageReq (MessageId, RestartMarker?)>

<!ATTLIST WriteMessageReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!-- MessageContent is transferred separately -->

<!ELEMENT WriteMessageRsp (Status, Error?)>
```

CommitMessage

```
<!ELEMENT CommitMessageReq (MessageId)>

<!ATTLIST CommitMessageReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED
```

```
     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!ELEMENT CommitMessageRsp (Status, Size, Error?)>
```

OpenMessage

```
<!ELEMENT OpenMessageReq (MessageId | (LocationName, Filename?, ClassId?, Status?), OutputName?, Mode?, Compress?, Eol?, View?, Fields?)>

<!ATTLIST OpenMessageReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!ELEMENT OpenMessageRsp ((Status, Error) | (MessageId, MessageType?, Sender?, Recipient?, TimeStamp?, IBTimeStarted?, Size?, Status?, InputName?, Filename?, OutputName?, ClassId?, ContentType?))>
```

ReadMessage

```
<!ELEMENT ReadMessageReq (MessageId, Size)>

<!ATTLIST ReadMessageReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!ELEMENT ReadMessageRsp (Status, Error?)>

<!-- MessageContent is transferred separately -->
```

CloseMessage

```
<!ELEMENT CloseMessageReq (MessageId, OutputName?)>

<!ATTLIST CloseMessageReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!ELEMENT CloseMessageRsp (MessageId, Status, Error?)>
```

AbortMessage

```
<!ELEMENT CloseMessageReq (MessageId)>

<!ATTLIST CloseMessageReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!ELEMENT CloseMessageRsp (MessageId, Status, Error?)>
```

GetLocationRights

```
<!ELEMENT GetLocationRightsReq (LocationName, IsRecipient?)>

<!ATTLIST GetLocationRightsReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED
```

```
     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>

<!ELEMENT GetLocationRightsRsp (LocationName Rights, Error?)>
```

ResendMessage

```
<!ELEMENT ResendMessageReq (MessageId)>

<!ATTLIST ResendMessageReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>
<!ELEMENT ResendMessageRsp (Status, Error?)>
```

RedirectMessage

```
<!ELEMENT RedirectMessageReq (MessageId, Recipient, Filename?)>

<!ATTLIST RedirectMessageReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>
<!ELEMENT RedirectMessageRsp (Status, Error?)>
```

MessageReceived

```
<!ELEMENT MessageReceivedReq (MessageId, Status, OutputName?, ErrorText?)>

<!ATTLIST MessageReceivedReq

     User CDATA #REQUIRED

     AuthMethod CDATA #IMPLIED

     Password CDATA #IMPLIED

     Response CDATA #IMPLIED

     Authzid CDATA $IMPLIED
     AccessClass CDATA #IMPLIED
     Version CDATA #IMPLIED>
<!ELEMENT MessageReceivedRsp (Status, Error?)>
```

GetNonPrivGlobalFlags

```
<!ELEMENT GetNonPrivGlobalFlagsReq >

<!ELEMENT GetNonPrivGlobalFlagsRsp (Status, FIPSOnlyTransport)>
```

GetLocationList

```
<!ELEMENT GetLocationListReq (LocationName, Type?, State?, PageNum?, PageLen?, Fields?)>

<!ATTLIST GetLocationListReq

          User CDATA #REQUIRED

          AuthMethod CDATA #IMPLIED

          Password CDATA #IMPLIED

          Response CDATA #IMPLIED

          Authzid CDATA $IMPLIED

          AccessClass CDATA #IMPLIED

          Version CDATA #IMPLIED>

<!ELEMENT GetLocationListRsp (PageNum, PageCount, Locations, Error?)>
```

CreateLocation

<!ELEMENT CreateLocationReq (LocationName)>

<!ATTLIST CreateLocationReq

User CDATA #REQUIRED

AuthMethod CDATA #IMPLIED

Password CDATA #IMPLIED

Response CDATA #IMPLIED

Authzid CDATA $IMPLIED

AccessClass CDATA #IMPLIED

Version CDATA #IMPLIED>

<!ELEMENT CreateLocationRsp (Status, Error?)>

DeleteLocation

<!ELEMENT DeleteLocationReq (LocationName)>

<!ATTLIST DeleteLocationReq

User CDATA #REQUIRED

AuthMethod CDATA #IMPLIED

Password CDATA #IMPLIED

Response CDATA #IMPLIED

Authzid CDATA $IMPLIED

AccessClass CDATA #IMPLIED

Version CDATA #IMPLIED>

<!ELEMENT DeleteLocationRsp (Status, Error?)>

RenameLocation

<!ELEMENT RenameLocationReq (LocationName, NewLocationName)>

<!ATTLIST RenameLocationReq

User CDATA #REQUIRED

AuthMethod CDATA #IMPLIED

Password CDATA #IMPLIED

Response CDATA #IMPLIED

Authzid CDATA $IMPLIED

AccessClass CDATA #IMPLIED

Version CDATA #IMPLIED>

<!ELEMENT RenameLocationRsp (Status, Error?)>

MarkMessage

<!ELEMENT MarkMessageReq (MessageId, Status)>

<!ATTLIST MarkMessageReq

User CDATA #REQUIRED

AuthMethod CDATA #IMPLIED

Password CDATA #IMPLIED

Response CDATA #IMPLIED

Authzid CDATA $IMPLIED

AccessClass CDATA #IMPLIED

Version CDATA #IMPLIED>

<!ELEMENT MarkMessageRsp (Status, Error?)>