

Integration Services Technical Overview

Copyright © 2017 by the State of Minnesota
State Court Administrator's Office
All Rights Reserved

Table of Contents

1. PREFACE	4
2. MESSAGING.....	4
2.1. Message Exchange Patterns	4
2.1.1. Request/Reply.....	4
2.1.2. Submission/Response.....	4
2.1.3. Publish/Subscribe	4
2.2. Message Format.....	4
2.3. Message Transport	5
3. MESSAGE FORMAT.....	5
3.1. XML Prefixes and Namespaces.....	5
3.2. SOAP Envelope	6
3.3. SOAP Headers	7
3.3.1. Addresses	8
3.3.1.1. Websphere MQSeries Addresses	8
3.3.1.2. Web Service Addresses.....	9
3.3.2. Headers of Type Endpoint Reference.....	9
3.3.3. Headers of Type Security/UsernameToken	9
3.3.4. Headers of type HoldResponse	10
3.4. SOAP Body	10
3.4.1. CourtXML Versioning	10
3.4.2. SOAP Faults	11
4. USING INTEGRATION SERVICES.....	14
4.1. Choosing an Environment.....	14
4.2. Sending Messages (Request and Submission).....	14
4.2.1. IBM Websphere MQ Series	14
4.2.2. Web Services	15
4.3. Receiving Messages	16
4.3.1. IBM Websphere MQ Series	16
4.3.1.1. Reply and Response Messages	16
4.3.1.1. Notification Messages.....	16

4.3.2.	Web Services	17
4.3.2.1.	Reply Messages.....	17
4.3.2.2.	Response Messages	17
4.3.2.3.	Notification Messages.....	17
4.3.2.4.	Receiving Pushed Messages	17
4.3.2.5.	Pulling Messages.....	18
4.3.2.5.1.	PullRequest.....	19
4.3.2.5.2.	PullReply.....	19
4.3.2.5.3.	ReleaseRequest.....	19
4.3.2.5.1.	ReleaseReply.....	20
5.	SECURITY	20
	APPENDIX A: ERRORS	21
	APPENDIX B: EXAMPLE SOAP FAULT ERRORS.....	22
	Examples with Invalid Code Values	22
	Example with Missing Required Element.....	25
6.	DOCUMENT REVISION HISTORY	26

1. Preface

This document gives a technical overview of the Integration Services provided by the Minnesota Supreme Court. From now on these services will be referred to as “Integration Services” in this document. It provides general details about how Integration Services work, and how to access them.

2. Messaging

2.1. *Message Exchange Patterns*

The Integration Services are implemented as sets of message exchanges. These exchanges all are based on one of the following 3 patterns.

2.1.1. Request/Reply

A request message is sent to the service and a reply message is returned to an address specified in the request message. Services that provide query access are developed using this pattern.

2.1.2. Submission/Response

A submission message is sent to the service. This submission may result in one or more response messages being returned to an address specified in the submission message. This category is used most often by services that are used to make updates to one of the Courts systems (EFile Submission). The purpose for response messages is usually to inform the submitter of the processing state of one of their submissions (Accepted, Rejected or Pending). Response messages can either be pushed or pulled (see section 5.3).

2.1.3. Publish/Subscribe

A message (referred to as a Notification message) is published by a service. Consumers that have previously subscribed to that message will have it sent to an address specified when their subscription was created. This category is most often used to publish business events to interested consumers. Response messages can either be pushed or pulled (see section 5.3).

2.2. *Message Format*

Messages are encoded using utf-8. These messages primarily consist of ASCII text characters, but may include non-ASCII characters if the source system supports them.

Unless otherwise noted, messages transmitted to and from the Courts Integration Services are formatted as SOAP messages. The body of each message is defined by a CourtXML message schema. See section 4 for more details on how SOAP messages are formatted. See the

documentation for the specific service you are using for information on the structure of the body of the message.

Note: depending on the toolset that you are using you may not need to even be aware that SOAP is being used. Your development tool may take care of building all of the soap elements and headers for you.

2.3. Message Transport

Integration Services are supported over multiple communications transports. These include

- Web Services
- IBM Websphere MQSeries (both Server and Client). Note: many software tools (such as Oracle and BizTalk) provide gateways to MQ Series and can use the MQ Series client to communicate with Queues in a MQ Series server environment.

See section 5 for more details on how to use each of the transports.

All messages are required to be transmitted over secure communication channels. This is done using SSL (HTTPS or MQSeries Channels using SSL). See the document [Integration Setup Procedures](#) for more information on setting up each transport.

3. Message Format

3.1. XML Prefixes and Namespaces

The following table lists the prefixes and xml namespaces that are used by integration services and are referenced throughout this document.

Prefix	XML Namespace	Specification(s)	Link/Description
soap*	http://schemas.xmlsoap.org/soap/envelope/	SOAP 1.1	http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
soap*	http://www.w3.org/2003/05/soap-envelope	SOAP 1.2	http://www.w3.org/TR/2003/REC-soap12-part1-20030624/
Wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd	WS-Security	http://schemas.xmlsoap.org/specs/ws-security/ws-security.htm
Wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing	WS-Addressing	http://www.w3.org/TR/ws-addr-core/
Wse	http://schemas.xmlsoap.org/ws/2004/08/eventing	WS-Eventing	http://www.w3.org/Submission/WS-Eventing/
	http://www.courts.state.mn.us/CourtXML/X.X.X (X.X.X corresponds to the version level)	CourtXML	http://www.mncourts.gov/?page=1368
Is	http://www.courts.state.mn.us/IS	Integration Services	Used as a namespace for fault codes specific to Courts Integration Services

Prefix	XML Namespace	Specification(s)	Link/Description
Xs	http://www.w3.org/2001/XMLSchema	XML Schema	http://www.w3.org/XML/Schema

* Services that fall under the Request/Reply and Submit/Response patterns support both SOAP 1.1 and SOAP 1.2. Reply and Response messages will use the same version of SOAP as their corresponding Request and Submission messages. Messages using the Publish/Subscribe pattern are sent using SOAP version 1.2.

3.2. SOAP Envelope

The root element of a SOAP message (either version 1.1 or 1.2) is the Envelope. The Envelope element contains a header section and a body section. The header section is used to hold message extensions (also referred to as message headers). Message headers contain information that controls how messages are processed. The body section is used for application-specific data and is considered the payload of the message. This is where the information that is the actual purpose for sending the message resides.

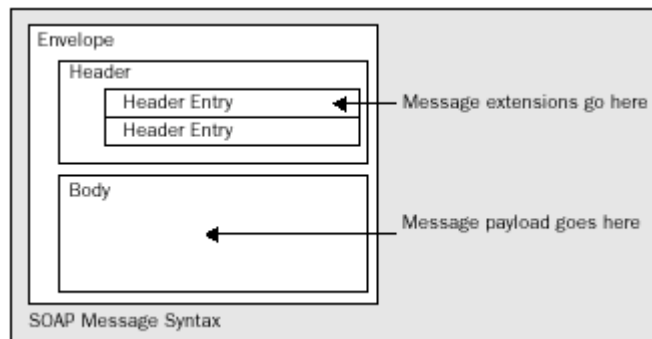


Figure 1: SOAP Message Structure

```

<?xml version="1.0"?>
<soap:Envelope xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    .....
  </soap:Header>
  <soap:Body>
    .....
  </soap:Body>
</soap:Envelope>

```

Figure 2: SOAP Message Syntax

3.3. SOAP Headers

The header section of the SOAP envelop contains XML elements known as message headers. These headers contain control information that is used to determine how to process the message. The following is a list of the most common headers that may be included in Integration Service messages. See their associated specifications (wse, wsse, wsa, ...), and the sample messages, for details on their structure.

HeaderElement	Description	Type
wsa:MessageID	Contains a unique identifier for this given message. All messages require this element with the exception of those submitted to the Integration Service using IBM Websphere MQSeries. If a message is received using IBM Websphere MQSeries and it doesn't have a wsa:MessageID header the value of the MQ Series messages MessageID property will be used.	xs:anyURI
wsa:RelatesTo	Contains the unique identifier (wsa:MessageID) from another message to which this message is somehow related. This header is used in Reply and Response messages to allow receivers to link up Reply messages with Request messages and a Response messages with Submission messages.	xs:anyURI
wsa:Action	Contains a string that uniquely identifies the given message. See the Integration Services Documentation for each service for the specific values that will be included in Reply, Response and Notification messages and that need to be included in Request and Submission messages.	xs:anyURI
wsa:To	Contains either an address to which the message is being delivered. Request and Submission messages should always contain the value http://www.courts.state.mn.us/IS/02 .	xs:anyURI
wsa:ReplyTo	Contains an address to which reply and response messages should be returned.	wsa:EndpointReference
is:ResponseTo	This header is used with e-File submission messages and contains the address that the response message(s) should be (asynchronously) delivered when using HTTP. This header is unnecessary when using MQ Series. This is necessary with HTTP because the wsa:ReplyTo header is used for the synchronous reply that indicates we have received the submission.	wsa:EndpointReference
is:HoldResponse	This header is used with e-File submission messages and is used to indicate that response messages should be held so they can be pulled at a later time by the submitter.	HoldResponse
wsa:FaultTo	Contains an address to which should be used to return fault messages should be returned. This header is optional and the ReplyTo header will be used in its absence.	wsa:EndpointReference

HeaderElement	Description	Type
wsse:Security/ wsse:UsernameToken	Contains information that will allow the receiver of a message to authenticate its sender. See the section called 'Headers of Type Security/UsernameToken' for more information on this header. This header is required in all Request and Submission (inbound to the court) messages and can optionally be configured to be included in Reply, Response and Notification messages (outbound from the court).	UsernameToken
wse:Identifier	Included in Notification messages (Publish/Subscribe). It contains an identifier that corresponds to the subscription that is being served by the delivery of a notification message. This identifier was assigned to the subscription when it was initially created.	xs:anyURI
wse:Expires	Included in Notification messages (Publish/Subscribe). If the subscription that is being serviced by this notification message is set to expire, this header will contain the expiration date. There are 2 reasons that a subscription could be set to expire: <ul style="list-style-type: none"> • It is for a version of CourtXML that is not current and an expiration date for that version has been set. • An expiration date was specified when the subscription was created. When this date passes no more notification messages will be delivered based on this subscription. This is a mechanism to alert the consumer that their subscription is about to end.	DateTime

3.3.1. Addresses

Addresses are used in messaging headers to identify endpoints to which messages should be routed. Integration Services are accessible via multiple transport mechanisms, each with a different format for how addresses are represented.

3.3.1.1. Websphere MQSeries Addresses

When using Websphere MQSeries as the transport, the addresses are formatted as follows:

`wmq:QUEUENAME@QUEUEMANAGER`

The first 4 characters are always 'wmq:'. The rest of the address is made up of 2 components separated by an "@" character. The first component is a queue name. The second component is the queue manager on which the queue resides.

```
<wsa:Address>wmq:MYQUEUE@MYQUEUEMANAGER</wsa:Address>
```

Figure 3: Sample wsa:Address element formatted for Websphere MQSeries

3.3.1.2. Web Service Addresses

Accessing Integration Services through Web Services uses HTTPS as the transport. HTTPS addresses are formatted as follows:

`https:webaddress`

The first 6 characters are always 'https:' followed by the rest of the characters that make up the web address.

```
<wsa:Address>https://myaddress.asmx</wsa:Address>
```

Figure 4: Sample wsa:Address header for HTTPS

3.3.2. Headers of Type Endpoint Reference

ReplyTo and FaultTo headers are of type Endpoint Reference and are used to specify the endpoint that should be used for reply and response messages. They will consist of a root element (named either wsa:ReplyTo or wsa:FaultTo) and the following child elements:

wsa:Address: Required, contains an address (see the section called 'Headers of Type Address').

wsa:ReferenceProperties: Optional, contains XML elements that the requestor/submitter wants returned as headers in reply/response messages.

```
<wsa:ReplyTo>
  <wsa:Address>wmq:MYQUEUE@MYQUEUEMANAGER</wsa:Address>
  <wsa:ReferenceProperties>
    <myafx:MyElement1 xmlns:myafx="mynamespace">MYDATA </myafx:MyElement1>
    <myafx:MyElement2 xmlns:myafx="mynamespace">THEDATA</myafx:MyElement2>
  </wsa:ReferenceProperties>
</wsa:ReplyTo>
```

Figure 5: Sample wsa:ReplyTo header

3.3.3. Headers of Type Security/UsernameToken

Headers of type Security/UsernameToken are used by the Integration Service to identify and authenticate the system that is consuming the Integration Service. They contain a Username element and a Password element.

For messages that are sent as input to Integration Services the Username and Password values will have been assigned to you at the time your configuration was setup as a consumer of Integration Services. If you need to access multiple environments separate accounts will be configured for Production, Development and QA. The passwords for Dev and QA will usually be the same while the password for Production

will be different. The Username is not specific to a certain user of your system, but is one value that corresponds to all users of your system when accessing Integration Services. Integration services use these values to determine what rights the sender has and to confirm that they have access to the particular service.

For messages output from Integration Services the values for Username and Password will be those that you specified when you requested access to integration services. The same values will be used no matter which Integration Service is sending you a message. You can use these values to confirm that the messages received come from a trusted source.

```
<wsse:Security>
  <wsse:UsernameToken>
    <wsse:Username>MyUsername </ wsse:Username>
    <wsse:Password>MyPassword</ wsse:Password>
  </wsse:UsernameToken >
</wsse:Security>
```

Figure 6: Sample wsse:Security/wsse:UsernameToken header

When using HTTP transport the components that you use to produce the UsernameToken elements will include many other elements and attributes that are not shown here.

At this time passwords are passed as plain text. This is one of the reasons that we require that all communication channels are set up using SSL encryption.

3.3.4. Headers of type HoldResponse

Headers of type HoldResponse are used within e-File submission messages to indicate that any resulting response messages should be held so they can be pulled at a later time by the submitter. It contains one child element named RetrievalCode This element is used to specify a value that is used to identify the message(s) that are being pulled. See section 5.3.2.5.

3.4. SOAP Body

The body of the SOAP envelope is where the actual payload of the message will be contained. The content is in XML format and its elements and structure are defined by the CourtXML message schema for the service that is being accessed. See the [Court Integration Services](http://www.mncourts.gov/is) website at <http://www.mncourts.gov/is> for a message schema for each service and a description of the message body for that service.

3.4.1. CourtXML Versioning

Each service will have its own schema which is based on the base CourtXML schema. These schemas will be versioned using the CourtXML namespace version concatenated to the schema version. For example: the first CaseGet schema developed using

CourtXML version 3 will be named CaseGet_3_1.xsd. If we need to produce a second version of the CaseGet schema, while still using the version 3 CourtXML base schema, it would be named CaseGet_3_2.xsd.

The version of the CourtXML schema to which the body of a given message corresponds will be shown by the value of the schemaVersion attribute of the root element (first child of the soap:Body element). An example is:

```
<MessageRootElement schemaVersion="3:1">
```

This version indicates that the associated message is at version 3 of CourtXML and version 1 of the specific message.

To support version migrations, Integration Services will support multiple versions at the same time. So, for instance, if version 3:2 of the CaseGet schema is published, then existing messages will continue to be serviced at the 3:1 version for a specified period of time. This will allow multiple partners to migrate to the new version of CourtXML as they are able, rather than be required to all migrate simultaneously. We plan to keep old versions of schemas available for at least 6 months after a new version is published in production.

Eventually old versions will expire and will no longer be supported. When this happens, any messages that are at that version will be rejected.

3.4.2. SOAP Faults

If some kind of error has occurred as a result of a request or submission message, the response and reply message bodies could consist of a SOAP fault. A common reason for receiving a SOAP fault is that the body of the corresponding request or submission message did not conform to an active version of the associated schema. Another is that the sender doesn't have the required access rights to use that service.

```

<soap:Fault>
  <soap:Code>
    <soap:Value>soap:Sender</soap:Value>
    <soap:Subcode>
      <soap:Value>soap:InvalidMessage</soap:Value>
    </soap:Subcode>
  </soap:Code>
  <soap:Reason>
    <soap:Text xml:lang="en">Message does not conform to schema.</soap:Text>
  </soap:Reason>
  <soap:Node>soap:Body</soap:Node>
  <soap:Role>http://www.w3.org/2003/05/soap-
envelope/role/ultimateReceiver</soap:Role>
  <soap:Detail>
    <is:DetailString>The element 'CaseGetRequest' in namespace
'http://www.courts.state.mn.us/CourtXML/2.0.0' has invalid child element 'Selectionx' in
namespace 'http://www.courts.state.mn.us/CourtXML/2.0.0'. List of possible elements expected:
'Selection'in namespace 'http://www.courts.state.mn.us/CourtXML/2.0.0'.</is:DetailString>
  </soap:Detail>
</soap:Fault>

```

Figure 7: Sample soap:Fault

The following elements make up a soap 1.2 fault:

Element	Description
soap:Code/ soap:Value	Code that shows where the error originated. Possible Values are: soap:Sender and soap:Receiver. The value soap:Sender indicates that the message had some problem with it prior to transmission and will have to be fixed prior to transmitting it again. It would not be appropriate to just retransmit the same message after receiving a soap:Sender fault. The value soap:Receiver indicates that a problem occurred after receiving the message and that the problem was not with the message itself. Retransmitting a message after a soap:Receiver message has been received may be successful.
soap:Code/ soap:Subcode/ soap:Value	Code that further describes the reason for the fault.
soap:Reason/ soap:Text	Textual description for the fault.
soap:Node	The node within the message from which the error resulted.
soap:Role	URI indicating the role of the process that is returning the soap fault.
soap:Detail	Element containing some additional information that should help with resolving the fault.

The wsa:Action element for messages that contain SOAP faults will be the following:

<http://schemas.xmlsoap.org/ws/2004/08/addressing/fault>

4. Using Integration Services

4.1. *Choosing an Environment*

Integration Services are accessible in 3 different environments (Development, QA/Test and Production). The service page for each service will provide you with the endpoint, or addressing information, for that service for each environment. The Development and QA/Test environments can be used as a part of your development and testing process. They are generally used for the following purposes, though depending on what services you are using, and maintenance schedules for those environments, you may be asked to use one or the other:

- **Development:** Development and testing using newly created integration services. These are services that we may be currently developing, or that are finished though haven't been migrated to production.
- **QA/Test:** Development and testing using existing services. This environment is usually at the same level as production and should be more stable than Development since code that it is running has been through the full testing cycle that is required for migration to production.

4.2. *Sending Messages (Request and Submission)*

Request and Submission messages need to be formatted as SOAP envelopes with all of the appropriate SOAP Headers. See section 4 for information on how to format the envelope and headers. For the `wsa:Action` header use the value that is specified in the service documentation for the particular service's request or submission message.

The body of the message is formatted according to the CourtXML message schema for the particular service that is being used.

The format for the message is the same regardless of the transport, but the methods you use to send them will vary depending on the transport chosen.

4.2.1. **IBM Websphere MQ Series**

When sending messages using IBM Websphere MQ Series your application will be responsible for formatting the message, including the soap envelope, headers and body as defined by the services message schema, and then placing that message on the Queue and Queue Manager specified as the MQ Series endpoint for the environment in which you are working. If you are using the MQ Series Client software you will be placing the message directly on that queue. If you have MQ Series Server installed you may be placing your message on a Remote Queue Definition that is linked to that input queue, or directly on that queue by referencing the specified queue manager as the "Remote" queue manager.

Within the message you will need to include a `wsa:ReplyTo` header that specifies the queue to which the reply message should be returned. If you are using the MQ Series Client we will have provided a queue for you to use for this purpose. If you are using MQ Series Server you will specify your queue manager and queue name. See section 4.3.2 for information on how to format this header.

See the [Integration Setup Procedures](#) document for information on how to set up connections to Integration Services using MQ Series and MQ Series Clients.

4.2.2. Web Services

Web Services provide a way to send Request and Submission messages to the Court Integration Services using HTTP. Court Integration Services can be accessed through 1 of 2 different web service endpoints. The only difference between them is with the WSDL that is presented by the service.

- SOAP Service Endpoint: WSDL does not specifically define the body of the message other than that it has to be formatted as XML.
- XML Web Service Endpoint: WSDL includes the appropriate CourtXML schema elements for the particular service you are using.

Messages are formatted the same regardless of which endpoint is used. The primary difference is with what your development tools may do for you based on what is in the WSDL document.

Messages can be sent to the web services in a variety of ways. They include:

- Referencing the SOAP Service
 - Create a web reference to the SOAP Service endpoint.
 - You should not have to do anything with creating soap envelopes or soap headers (other than adding the `wsse:Security` header).
 - Manually create the body of the message based on the CourtXML message schema.
 - Send the body of the message to the service by calling a method of the service, passing the body as a parameter.
- Referencing the XML Web Service
 - Create a web reference to the XML Web Service endpoint.
 - You should not have to do anything with creating soap envelopes or soap headers (other than adding the `wsse:Security` header).
 - Make use of the set of classes generated from the WSDL to create the request or submission object.
 - Call a method of the service to send the request or submission to the service passing the object as a parameter.
- Use an HTTP Post to send the message
 - Manually create the full soap envelope, including all headers (as described in section 4) and the body as defined by the services CourtXML message schema.

- Send the message to either the SOAP Service or XML Web Service endpoints using a HTTP Post and with the following HTTP headers set:
 - SOAPAction: the same value as in the wsa:Action header of the message.
 - Content-Type:text/xml;charset=utf-8

4.3. Receiving Messages

Messages received will be formatted as SOAP envelopes with all of the appropriate headers. See section 4 for information on how the SOAP envelope and headers will be formatted.

The body of the message will be formatted according to the CourtXML message schema for the particular service that is being used.

The format for the message is the same regardless of the transport, but the methods you use to receive them will vary depending on the transport chosen.

4.3.1. IBM Websphere MQ Series

All messages published to MQ Series queues will be written using code page 1208 to support utf-8 encoding of data.

4.3.1.1. Reply and Response Messages

Reply and Response messages will be sent to the Queue and Queue Manager that were specified in their corresponding request and submission message's wsa:ReplyTo header. If a reply or response is a SOAP fault then the values provided in the wsa:FaultTo header will be used if provided. Reply and Response messages can be correlated with their associated Request and submission messages in the following ways:

- The reply or response MQ Series message's CorrelationID property will be the same as the MessageID property of the request or submission message.
- The wsa:RelatesTo SOAP header of the reply or response message will be the same as the wsa:MessageID SOAP header of the request or submission message.
- Any elements included in the request or submission messages (wsa:ReplyTo|wsa:FaultTo)/wsa:ReferenceProperties element will be returned as headers in the reply or response message (see section 4.3.2).

4.3.1.1. Notification Messages

Notification messages will be sent to the Queue and Queue Manager that were specified when the subscription was initially created. Notification messages will contain a header wse:Identifier that will contain the identifier for the subscription for which the notification is being sent. If a subscription has an expiration date set its notification messages will also contain a header wse:Expires which will contain the date and time that that subscription will stop receiving messages.

4.3.2. Web Services

4.3.2.1. Reply Messages

Reply messages, unless otherwise specified, are returned synchronously as the return value from the method call that submitted the request. Nothing additional needs to be done to receive these reply messages. If the specific Request/Reply service that you are using provides an asynchronous option the reply messages will be treated the same as response messages.

4.3.2.2. Response Messages

For response messages, when the initial submission message is sent there will be a synchronous response (return from the submission method call) that indicates whether the message was received or not. If the message could not be received for any reason that response will be a SOAP fault. All other response messages that are sent as a result of a submission will be returned asynchronously (at a later time). These asynchronous response messages can either be pushed to a web service whose URL was specified in the submission messages is:ResponseTo header, or pulled. See the specific documentation for the service you are using for how to specify which method (push or pull) you want to use for receiving these responses. See sections 5.3.2.4 and 5.3.2.5 for information on how to receive pushed and pulled messages.

4.3.2.3. Notification Messages

Notification messages, when triggered, can either be pushed to a web service whose URL was configured on the associated subscription at the time that the subscription was created, or pulled. See the specific documentation for the service you are using for how to specify which method (push or pull) you want to use for receiving these messages. See sections 5.3.2.4 and 5.3.2.5 for information on how to receive pushed and pulled messages.

4.3.2.4. Receiving Pushed Messages

Pushed messages are sent to an endpoint specified by the consumer (either via the is:ResponseTo header of corresponding request and submission messages, or via the configuration of the notification subscription) at the time the messages are produced.

To receive Pushed messages through web services you will need to provide a message receiver web service. That service will need to have the following characteristics:

- Be accessible via HTTPS (using SSL for data encryption while being transmitted).
- Include a method that will be used to receive the message that has the following characteristics:
 - Have its assigned SOAP action equal to the message's Push SOAP Action as identified in the specific service's documentation for that message.
 - It must have a binding of "document" and a use of "literal".

- It must have both input and output, and return a fault if the message could not be received. For successfully received messages it doesn't matter what the output is as long as it is not a fault. The output for a successfully received message could be an empty soap envelope, or a value that indicates that the message was received.
- It is recommended that this method do nothing more than store the message at a location where it can be processed at a later time.
- If the service that is pushing messages could include SOAP fault messages then the receiver web service must also include a method to receive those faults. That method should have the same characteristics as defined in the previous bullet and be associated with the SOAP Fault action (<http://schemas.xmlsoap.org/ws/2004/08/addressing/fault>).

4.3.2.5. PullingMessages

Pulled messages are held by the Court Integration Broker at the time that they are produced so that they can be pulled by the consumer at a later time. They are requested to be held by including specific headers in Request and Submission messages or via the configuration of the notification subscription. Pulling messages is appropriate for smaller offices where a web server, web service and digital certificates required to receive messages may not be available, and where volumes are low and some additional latency is not an issue.

To receive Pulled message there are 2 additional Request and Reply messages that are used. Each service that supports Pulled messages will provide both a Pull request and reply and a Release request and reply in the services message schema. See the associated schema for the message formats and the associated service documentation for the SOAP Actions that are associated with the Pull and Release requests.

For each message that was held by the courts integration broker for the consumer, the consumer will need to do a pull request to get the message. Once the message has been pulled and processed successfully they will need to do a Release request to inform the Court Integration Broker that it can dispose of that message. Note: subsequent Pulls without doing a Release will always result in the same message being pulled. This is done to assure that messages are not lost if system problems occur. The following algorithm can be used to pull messages:

1. Submit a Pull request
2. If a message is returned in the reply:
 - a. Process that message
 - b. Submit a Release request to dispose of that message
 - c. If the reply from the pull request indicates that there are more messages to be pulled repeat starting at step 1.

If an error occurs while processing the message and a release is not done, the next time the pull request is initiated that message will be returned again.

It is recommended that consumers perform pull requests for messages at a minimum of 1 time each day, and a maximum of every hour.

The following describe each of the Pull/Release messages:

4.3.2.5.1. PullRequest

This message is used to initiate the pull of a held message and to specify the selection criteria for the message that is to be pulled. The selection criteria consists of a code value named RetrievalCode. This value has to match the value that was included in the associated Submission message, or that is configured on the notification subscription. The value for RetrievalCode that is chosen is completely up to the consumer and depends on how the consumer wants to pull messages. Some examples of how it can be used are:

- Use the same value for all messages of the type that is being pulled (this is how it works for Notification messages).
- Use a different value for each message.
- Use the same value for each message for a given department.

4.3.2.5.2. PullReply

This message is the reply to a PullRequest. It will include the following information:

- The RetrievalCode that was used in the request.
- The number of held messages that had the same RetrievalCode that are still available to be pulled (not including the current message).
- If a message was found that matched the RetrievalCode it will also include the following:
 - An identifier for the pulled message (MessageID) to be used later in the ReleaseRequest.
 - A count of how many times this message has been pulled (MessagePulledCount). This value usually will be 1 though may be greater than 1 if errors have occurred during prior pulls which caused it not to be released.
 - An element containing the message that is being pulled. See the service documentation and schema for the format for this element. If the service that you are using could publish faults then this element could also be a SOAP fault.
- Any additional soap headers that are specified to be included with the message that is being pulled will be included as headers in the PullReply. Some examples of this are EFileID and SubmittingAgencyORI of EFile response messages, and the wse:Expires and wse:Identifier headers of notification messages.

4.3.2.5.3. ReleaseRequest

This message is used to inform the Court Integration Broker that a specific held message can be disposed of so that the next pull request will retrieve the next message. It

includes the MessageID value that was retrieved in a prior PullReply message that identifies the message that can be disposed.

4.3.2.5.1. ReleaseReply

This message is used to inform the consumer that a message has been successfully released. It includes the MessageID value that was included in the ReleaseRequest, and an indicator that the release was successful.

5. Security

Integration Services use secure communication channels to protect messages as they are being transmitted between servers.

Each message that is either input to or output from an Integration Service contains credentials that identify the submitting system. At the time that you are registered for access to Integration Services you will be provided with the credentials that you will need to send along with messages that are input to Integration Services (Requests and Submissions), and will be able to specify the credentials that you want returned to you in Reply, Response and Notification messages. Currently these credentials are in the form of a user name and password. See the section called 'Headers of Type Security/UsernameToken' for a description of the format for these credentials.

For input messages, Integration Services use these credentials to determine the rights to attribute to the sender. These rights include at a minimum the permission to access the particular service. They may also include others, such as permissions to certain court locations, lines of business or confidential access.

Appendix A: Errors

The following table lists the soap faults that are common to integration services. See the document for the specific integration service that you are using for additional faults and errors that may be specific to that service.

#	Type	Error Code	Error Text	Description/Resolution
1	SOAP Fault	wsse:InvalidSecurityToken	An invalid security token was provided.	Confirm that the Username and Password that you specified are valid.
2	SOAP Fault	wsse:MissingSecurityToken	Missing Security Token.	Include a wsse:Security header with your assigned Username and Password.
3	SOAP Fault	wsse:UnauthorizedAccess	Consumer does not have authorization to use service.	Confirm that your Username was assigned the required rights.
4	SOAP Fault	soap:VersionMismatch	Cannot Determine Version Level.	Confirm that the message has the proper schemaVersion and xmlns namespace.
5	SOAP Fault	soap:VersionMismatch	An Unsupported CourtXML Namespace was provided.	Check on which versions of the message you are submitting are still active.
6	SOAP Fault	is:NotWellFormed	The Input Document is not well formed XML.	Check the formatting of your message. Confirm that it is well-formed XML.
7	SOAP Fault	soap:InvalidMessage	Message does not conform to schema. Refer to Appendix B for some examples.	Check the formatting of your message. Confirm that it matches the definition in the schema.
8	SOAP Fault	is:SystemError	<i>Some error message describing the technical error that has occurred.</i>	Some technical system problem is preventing the processing of the message. Report this problem to the Courts integration support.

Appendix B: Example SOAP Fault Errors

Messages that contain invalid code values are returned to the submitter as SOAP faults. These 'enumeration-type' errors indicate that something about a codified element value in the message is invalid. Either the code itself is not valid according to the enumeration in the corresponding CourtXML message schema, or, the use of the code in the context of the message is invalid according to the related simple type companion file.

Examples 1 and 2 below are examples of errors that indicate a problem with a value supplied in the message.

Schema validation errors are detected and formatted by the XML parser component of the .NET framework.

Examples with Invalid Code Values

Example 1:

```
- <soap:Fault>
- <soap:Code>
  <soap:Value>soap:Sender</soap:Value>
- <soap:Subcode>
  <soap:Value>soap:InvalidMessage</soap:Value>
</soap:Subcode>
</soap:Code>
- <soap:Reason>
  <soap:Text xml:lang="en">The
  'http://www.courts.state.mn.us/CourtXML/3:ID' element is invalid - The
  value '5000' is invalid according to its datatype
  'http://www.courts.state.mn.us/CourtXML/3:StatutoryChargingOffenseType' -
  The Enumeration constraint failed. LineNumber = 225, LinePosition =
  21</soap:Text>
</soap:Reason>
  <soap:Node>soap:Body</soap:Node>
  <soap:Role>http://www.w3.org/2003/05/soap-
  envelope/role/ultimateReceiver</soap:Role>
- <soap:Detail>
- <soap:Envelope xsi:schemaLocation="http://www.w3.org/2003/05/soap-
  envelope http://www.w3.org/2003/05/soap-envelope">
```

Diagram annotations:

- 1: Points to the error message text: 'The Enumeration constraint failed. LineNumber = 225, LinePosition = 21'
- 2: Points to the element name 'ID' in the error message: 'http://www.courts.state.mn.us/CourtXML/3:ID'
- 3: Points to the start of the reason text: '<soap:Text xml:lang="en">The'
- 4: Points to the error message text: 'The Enumeration constraint failed. LineNumber = 225, LinePosition = 21'
- 5: Points to the end of the error message text: '21'

1. 'The enumeration constraint has failed' indicates a schema validation error. The message failed because it included an invalid value for a codified element in the message schema. In CourtXML schemas, elements that represent codified values in MNCIS will have an *enumeration*, or list, of the values that are valid for that element.

2. This is the name of the element that contained the invalid value. In this example, the name of the element is *ID*.

3. This is the value from the submission message that is invalid according to the enumeration in the schema. In this example, the invalid value is *5000*.

4. This is the CourtXML type for which the supplied value is invalid. In this example, the type is *StatutoryChargingOffenseType*. The corresponding simple type companion file from which the list of valid values is derived for the schema enumeration is called *StatutoryChargingOffenseType.xml*.

Note that the enumeration in the schema will contain obsolete values. An obsolete value could be valid on an outbound notification from MNCIS, but it would not be valid on a submission into MNCIS. The simple type companion file identifies codes that are obsolete with an 'obsoleteDate' attribute.

5. This is the line number and position within the line, from the body of the submission message that contains the invalid value. It is somewhat approximate because it could vary slightly depending upon the tool that is being used to view the XML document. This line number/position information only takes into account the body of the message – it does not include any of the SOAP elements or headers.

Example 2:

This message passed 'schema validation', but failed on subsequent validation of the Community of Offense code done by the Integration Broker.

```
- <soap:Fault>
- <soap:Code>
  <soap:Value>soap:Sender</soap:Value>
- <soap:Subcode>
  <soap:Value>soap:InvalidMessage</soap:Value>
</soap:Subcode>
</soap:Code>
- <soap:Reason>
  <soap:Text xml:lang="en">Unknown Enumeration value 'Stillwater' from
  SimpleTypeCompanionFileCommunityOfOffenseTextTypefor
  court:MN027015J</soap:Text>
</soap:Reason>
  <soap:Node>soap:Body</soap:Node>
  <soap:Role>http://www.w3.org/2003/05/soap-
  envelope/role/ultimateReceiver</soap:Role>
- <soap:Detail>
- <soap:Envelope xsi:schemaLocation="http://www.w3.org/2003/05/soap-
  envelope http://www.w3.org/2003/05/soap-envelope">
```

1. 'Unknown enumeration' is again indicating a problem with the value in the message. In this example, the code itself is valid and does appear in the enumerated list of values for CommunityOfOffenseTextType. The problem is with the use of the code in conjunction with some other data in the message.
2. This is the code value that is deemed to be invalid in the message. In this example, the value is *Stillwater*.
3. This is the CourtXML type which has a value of 'Stillwater'. The corresponding simple type companion file is called CommunityOfOffenseTextType.xml.
4. The value 'MN027015J' provides the context for determining why a value of 'Stillwater' for the Community of Offense' is invalid for this message.

MN027015J is the court jurisdiction ORI to which the example case initiation message is being submitted. MN027015J represents Hennepin District Court. For a case being initiated in Hennepin County, 'Stillwater' is not a valid value for Community of Offense.

By examining the simple type companion file noted above (CommunityOfOffenseTextType.xml), it can be determined that 'Stillwater' is a valid Community of Offense value for the following court ORIs only: MN082015J (Washington Co District Court - Stillwater) and MN082025J (Washing Co District Court – Cottage Grove). Listed below is the relevant entry from the simple type file:

```

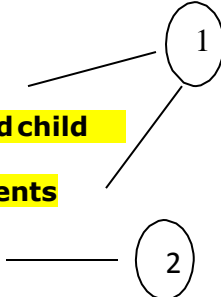
- <EnumerationValue code="82STILLW">
  <Text>Stillwater</Text>
- <AssociatedValue type="InternalID">
  <Text>56052</Text>
  </AssociatedValue>
- <AssociatedValue type="ProsecutingAgencyORI">
  <Text>MN062025A</Text>
  </AssociatedValue>
- <AssociatedValue type="ProsecutingAgencyORI">
  <Text>MN082013A</Text>
  </AssociatedValue>
- <AssociatedValue type="ProsecutingAgencyORI">
  <Text>MN082011A</Text>
  </AssociatedValue>
- <AssociatedValue type="CourtJurisdictionORI">
  <Text>MN082025J</Text>
  </AssociatedValue>
- <AssociatedValue type="CourtJurisdictionORI">
  <Text>MN082015J</Text>
  </AssociatedValue>
</EnumerationValue>

```


Example with Missing Required Element

Example 3:

```
- <soap:Fault>
- <soap:Code>
  <soap:Value>soap:Sender</soap:Value>
- <soap:Subcode>
  <soap:Value>soap:InvalidMessage</soap:Value>
</soap:Subcode>
</soap:Code>
- <soap:Reason>
  <soap:Text xml:lang="en">The element 'AddressUSNonStandard' in
  namespace 'http://www.courts.state.mn.us/CourtXML/3' has invalid child
  element 'AddressCity' in namespace
  'http://www.courts.state.mn.us/CourtXML/3'. List of possible elements
  expected: 'AddressLine1' in namespace
  'http://www.courts.state.mn.us/CourtXML/3'. LineNumber = 75,
  LinePosition = 10</soap:Text>
</soap:Reason>
  <soap:Node>soap:Body</soap:Node>
  <soap:Role>http://www.w3.org/2003/05/soap-
  envelope/role/ultimateReceiver</soap:Role>
- <soap:Detail>
- <soap:Envelope xsi:schemaLocation="http://www.w3.org/2003/05/soap-
  envelope http://www.w3.org/2003/05/soap-envelope">
```



1. This means that a required child element of AddressUSNonStandard was not included in the message document, or was not in the expected location in the document. The parser is encountering an element called AddressCity where it should have found AddressLine1. In the schema for this message, AddressLine1 is the first required child element when AddressUSNonStandard is used as the address structure for the defendant's address.

2. This is the line number and position within the line, from the body of the submission message that contains the invalid value. It is somewhat approximate because it could vary slightly depending upon the tool that is being used to view the XML document. This line number/position information only takes into account the body of the message – it does not include any of the SOAP elements or headers.

6. Document Revision History

Date	Author	Revision Highlights
10/28/2005	T. Buchholz	Created.
4/10/2006	T. Buchholz	Added SOAP actor.
9/11/2006	T. Buchholz	Migrated to SOAP version 1.2
4/25/2007	T. Buchholz	Added information regarding HTTPS transport.
7/12/2007	R. Gosewisch	Updated format.
2/11/2008	T. Buchholz	Add more information on Web Services access to Integration Services.
6/11/2008	R. Gosewisch	Added Appendix B to provide some examples of SOAP validation errors when there are invalid code values or missing elements in a message.
6/9/2009	T. Buchholz	Removed Soap Actor.
2/13/2017	R. Rowan	Modified format of this document. No content changes.