

HMRC NCTS XML Channel API

Version 1.1 – 11/03/08

Version History

<i>Version</i>	<i>Date</i>	<i>Author</i>	<i>Comments</i>
0.1	30/01/06	Alastair Barnes	Initial draft
0.2	07/02/06	Alastair Barnes	Updated after initial review
0.3	09/02/06	Alastair Barnes	Updated to include XML samples
0.4	16/02/06	Alastair Barnes	Updated to include HMRC header
0.5	20/02/06	Alastair Barnes	Updated to include Character encoding of EDIFACT messages
1.0	24/02/06	Alastair Barnes	First Issue
1.1	11/03/08	Lisa S, Neil P	Section 2.10 updated to make explicit an active enrolment is necessary and to include an authorisation error

Contents

1.	Introduction.....	4
2.	Major Aspects of the Interface.....	5
2.1	Transport and Security.....	5
2.2	Stateless Operation.....	5
2.3	Sessionless Operation.....	5
2.4	Error Handling.....	5
2.5	API Operation Overview.....	6
2.5.1	submitDocument.....	6
2.5.2	getMessages.....	6
2.5.3	getMessagesByTransactionId.....	6
2.5.4	getTransactionStatus.....	6
2.6	IE Request Submission.....	6
2.7	Transaction Identification.....	6
2.8	IE Response Message Indexing.....	6
2.9	IR Response Message Retrieval.....	7
2.9.1	Retrieval by Sequence Number.....	7
2.9.2	Retrieval by Transaction ID.....	8
2.10	Authentication.....	8
2.11	hmrcInfo Header.....	10
2.12	Character Encoding.....	12
3.	API Calls.....	14
3.1	submitDocument.....	14
3.2	getMessages.....	15
3.3	GetMessagesByTransactionId.....	19
3.4	getTransactionStatus.....	21
3.5	Supporting types.....	23
3.5.1	MessageHeader.....	23
3.5.2	ResponseMessage.....	23
3.5.3	ResponseSummary.....	24
4.	Appendix A – Example SOAP Messages.....	25
4.1	submitDocument.....	25
4.1.1	SOAP Request.....	25
4.1.2	SOAP Response.....	25
4.2	getMessages.....	26

4.2.1	SOAP Request	26
4.2.2	SOAP Response	26
4.3	getMessagesByTransactionId	27
4.3.1	SOAP Request	27
4.3.2	SOAP Response	27
4.4	getTransactionStatus	28
4.4.1	SOAP Request	28
4.4.2	SOAP Response	28
5.	Appendix B SOAP Fault Examples	29
5.1	SubmitDocument	29
5.1.1	Bad request.....	29
5.1.2	General processing exception	29
5.2	getMessages	30
5.2.1	Bad request.....	30
5.2.2	General processing exception	30
5.3	getMessagesByTransactionId	31
5.3.1	Bad request.....	31
5.3.2	General processing exception	31
5.4	GetTransactionStatus	32
5.4.1	Bad request.....	32
5.4.2	General processing exception	32
6.	Appendix B Notes on EDIFACT encoding	33

1. Introduction

The HMRC's NCTS XML Channel introduces a new reliable channel facilitating the exchange of NCTS Information Exchange EDIFACT messages. The agreed solution is based on providing direct access to HMRC via an XML channel (based on Web Services) and leveraging to a maximum extent existing EDIFACT processing components.

This solution has been coined as the "XML-Wrap" approach: the XML channel is limited to ensuring the reliable exchange of existing EDIFACT messages (already used in the email route) by wrapping them in an XML envelope.

This document defines the client view of the Application Programming Interface (API) for HMRC's NCTS XML Channel in abstract terms – that is, in terms of the data that passes between client applications and the service itself, and the modes of interaction. The exact programming will depend upon the SOAP toolkit (if any) employed in the construction of the client application – most will be able to present a remote method invocation interface given an appropriate Web Services Description Language (WSDL) file describing the service interface. However, there is sufficient detail in this document (and the WSDL file) to allow the low-level construction and handling of SOAP messages if necessary. The selected interaction style is Document Oriented, i.e. the SOAP message will contain a document and use literal encoding. The client-server model of interaction adopted for this solution is synchronous.

2. Major Aspects of the Interface

2.1 Transport and Security

The service will be accessible only via SOAP/HTTP over an SSL v3.0 / TLS connection (“secure HTTP” or HTTPS) to ensure the confidentiality of requests and returned data. The NCTS XML Channel will be implemented using SOAP v1.1, which will support v1.1 SOAP clients.

Client authentication will be based on Government Gateway credentials (user id and password). This will conform to the WS-Security (wsse) standards as detailed in ‘Web Services Security UsernameToken Profile 1.0, OASIS Standard 200401, March 2004’. Prior registration on the Government Gateway, thus obtaining a username and password combination will be required. Subsequent enrollment and activation of the NCTS Gateway service will also be necessary.

2.2 Stateless Operation

The NCTS XML Channel provides a stateless API for Trader client applications to invoke web service operations made available by the service. Essentially this means that each call to the interface is independent and contains enough information to invoke and complete the desired web service operation – no “state” or temporary information pertaining to any previous or subsequent operation invocations is held by the application server. This stateless operation improves resilience in the face of unexpected client or service failure.

2.3 Sessionless Operation

The core web service operations exposed in this API will be invoked in “sessionless” mode, with the provision of Government Gateway credentials (user id and password) required in the WS Security header in each call. The use of digital certificates, SAML or other tokens is not currently supported in this release of the NCTS XML Channel.

2.4 Error Handling

NCTS XML Channel-specific errors identified for each Web Service operation are raised by the application as custom errors, and communicated to the client application using the SOAP <Fault> element in the SOAP envelope. Access to SOAP faults will depend on the error handling provisions of the SOAP client library in use.

After making a NCTS XML Channel method call, the client application is expected to check for errors. If no errors of any kind are raised, the response is expected to contain the normal output from the call, and processing can continue. If errors were raised, further checks for custom errors can be made to determine the cause and, if necessary, the appropriate remedial action to be undertaken.

2.5 API Operation Overview

2.5.1 submitDocument

This operation provides the Trader client application with capability to send an XML message wrapping an Information Exchange (IE) EDIFACT message to the NCTS XML Channel to be processed. The SOAP response generated by this invocation contains a Transaction Identifier (see section 2.6 IE Response message Indexing)

2.5.2 getMessages

This operation provides the Trader client application with capability to retrieve the responses received by the system from the ECN/MCC after a specified Response Sequence Number from the NCTS XML Channel. The SOAP response generated by this invocation contains an array of IE response messages.

2.5.3 getMessagesByTransactionId

This operation provides the Trader client application with capability to retrieve the responses related to a specific Transaction (identified via the Transaction identifier) from the NCTS XML Channel. The SOAP response generated by this invocation contains an array of IE response messages.

2.5.4 getTransactionStatus

This operation provides the Trader client application with capability to get the status of a specific Transaction (identified via the Transaction Identifier) from the NCTS XML Channel. The SOAP response generated by this invocation contains an array of Response Summaries plus some information about the originating Request.

2.6 IE Request Submission

The submission of an NCTS EDIFACT IE Message is achieved by the invocation of the “submitDocument” web service operation. IE Request messages are communicated in their EDIFACT form, however in order to conform to SOAP standards the EDIFACT message must be UTF-8 encoded. The SOAP message has a single element, which contains the EDIFACT IE Message.

2.7 Transaction Identification

In the context of the NCTS XML Channel application a transaction is a set of IE messages constituted by an original Information Exchange message submitted by the Trader and its directly related Information Exchange responses. A unique identifier is generated for a Transaction and is sent back to the Trader as part of the “submitDocument” SOAP Response message. The Transaction Identifier is essentially used for message monitoring and tracking purposes.

2.8 IE Response Message Indexing

Each IE Response message received from ECN/MCC is associated with the user who submitted the originating IE Request message. This is the user who’s user ID was present in the WS Security header of the submitDocument web service operation call. Additionally each IE Response message is assigned a sequence number, which is unique to a user, however not globally unique within the NCTS XML Channel. This

allows the response sequence numbers to be contiguous for a user thus removing ambiguity regarding missing messages. Response sequence numbers are assigned by the NCTS XML Channel Database and have a scale of 9 digits (up to 999999999), which, based on current volumetrics, is sufficient to last many tens of years. All sequence numbers will start at 1 and monotonically increase (index number 0 is intentionally not used)

The uniqueness (across all IR Responses, for a specific user), contiguousness and chronological ordering of the sequence numbers are fundamental to the way the API works. IE Responses can be retrieved based on their sequence numbers and “highestReturned” – the “highestReturned” is the only piece of “state” that needs to persist on the client side (aside from credentials information) during the normal operation of the NCTS XML Channel interface.

The NCTS XML Channel system does not perform any re-ordering of IE Response messages once they have been received from ECN/MCC. Sequence numbers are assigned to the Responses as they arrive.

2.9 IR Response Message Retrieval

2.9.1 Retrieval by Sequence Number

Initially, before any call is made to the service, the Trader’s “highestReturned” is assumed to be zero (this is why 0 is not used as a data item index number). After a successful “getMessages” call (made by indicating that the caller “lastRetrieved” up to “sequence number” 0) the returned “highestReturned” will indicate the highest sequence number of the IE Response messages returned. This value should be retained and used in subsequent “getMessages” calls as the “lastRetrieved” value – only data items with higher sequence numbers than this will be returned, and again the “highestReturned” will be updated (this cycle is the crux of the “getMessages” interaction procedure over time).

In addition to the above there is an in-built message-size limit of the service. The “getMessages” operation will have a maximum number of IE Response messages defined for a single retrieval designed to keep message size below a pre-defined limit. In these cases, “getMessages” will result in the return of only some of the IE Response messages available (always the lowest sequence numbers). Where this occurs, a “moreAvailable” flag is set to “true” by the service and the returned “highestReturned” is set to the sequence number of the highest IE Response message actually returned.

There is also a mechanism by which the Trader may elect to impose a message-size limit (the “getMessages” operation will have an optional “maxResponses” which allows the enforcement of a maximum number of IE Response messages for a single retrieval). In these cases, “getMessages” will result in the return of only the number of the IE Response messages specified in “maxResponses” (always the lowest sequence numbers). As above, the “moreAvailable” flag is set to “true” if there are more IE Responses available and “highestReturned” is set to the sequence number of the highest IE Response message actually returned. If the results of a “maxResponses” limited “getMessages” operations exceeds the service in-built message size limit the latter will take precedence.

It would be desirable to test the “moreAvailable” in the normal course of events after every “getMessages” call in case there are more IE Response messages available. If there are, one or more further calls should be made, each time using the returned “highestReturned” and updating the “lastRetrieved” value on each call. In this manner, large downloads can be accommodated without prior arrangement.

2.9.2 Retrieval by Transaction ID

It will also be possible to retrieve responses by Transaction ID (for a description of transaction ID see section 2.5 Transaction Identifier). The “GetByTransactionIdRequest” is invoked by specifying Transaction Identifier of the transaction for which IE Responses are required. This will result in the return of only the IE Response messages related to transaction.

Due to the finite number of IE Response messages generated as the result of an IE Request message there is no limit, system imposed or otherwise, on the size of the result set for this operation.

2.10 Authentication

In order to provide secure access to the web services exposed by the NCTS XML channel Trader client applications must be authenticated before they can be allowed access to the web service operations. This will be achieved by utilising WS-Security (wsse) standards as detailed in ‘Web Services Security UsernameToken Profile 1.0, OASIS Standard 200401, March 2004’.

Header - wsse:Security			
Element - wsse:UsernameToken			
<i>Name</i>	<i>Type</i>	<i>Example Value</i>	<i>Comment</i>
Username	String	testuser1	Government Gateway user ID of the party invoking the Web Service operation
Password	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText	testpass1	The actual password for the user ID

The WS-Security specification supports two different methods of passing the password in the UsernameToken, either ‘PasswordText’ or ‘PasswordDigest’. The ‘PasswordText’ attribute allows the password to be included in the security token in clear text. It will be the only method supported by the NCTS XML Channel API since all message interactions are encrypted for transit as a result of using HTTPS.

In its simplest form the UsernameToken looks like this:

```
<wsse:Security>
  <wsse:UsernameToken>
    <wsse:Username> [Gateway-user-id] </wsse:Username>
    <wsse:Password Type="...#PasswordText"> [Password] </wsse:Password>
  </wsse:UsernameToken>
</wsse:Security>
```

The values in square brackets would be replaced by the actual user-id and the clear-text password. Note that the value of the Password element Type attribute has been shortened for clarity in the example – in full it should be:

<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText>

However, since this is the default value for the Type attribute it may be omitted.

The namespace for the conventional prefix ‘wsse’ is

<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd>

The UsernameToken should be placed inside a wsse:Security element within the header of a SOAP message. Recent SOAP toolkits that support WS-Security may provide explicit support for the addition of security tokens. In all other cases, an explicit insert of a wsse:Security section containing the UsernameToken into the SOAP header will be required by whatever means available.

The user must also have an active service enrolment on the Government Gateway for the NCTS service, and the call will fail if this is not the case.

Below are the reported error conditions resulting in the failure of the method call (custom errors raised as SOAP faults):

Authentication exception	
faultcode	fault:FailedAuthentication
faultString	[Security:090304]Authentication Failed: User <username> javax.security.auth.login.FailedLoginException: [Security:090302]Authentication Failed: User <username> denied
Cause	The Government Gateway credentials (username and password) provided in the <wsse:UsernameToken> element of the <wsse:Security > were invalid
Outcome	The authentication failure will be logged and this SOAP fault will be returned to the client. In the event that there are three consecutive failed authentication attempts with the same user name the account will be locked out for a three-hour period.
Remedial action	Invoke the operation with valid credentials.

Authorisation exception	
faultcode	fault:FailedAuthorisation
faultString	Authorisation Failed: User <username> does not have access to this service
Cause	The user was successfully logged into the Government Gateway but was subsequently found to be lacking an active service

	enrolment for NCTS
Outcome	The user will not be locked-out after a certain number of attempts (as they would with incorrect password)
Remedial action	The user needs to set-up the NCTS service enrolment on the Government Gateway

General processing exception	
faultcode	fault:FailedAuthentication
faultString	[Security:090304]Authentication Failed: User <username> javax.security.auth.login.FailedLoginException: [Security:090302]Technical Exception when performing Authentication
Cause	There was an internal problem performing the authentication operation and it did not complete successfully.
Outcome	The invocation of the operation has failed due to an internal processing exception during authentication.
Remedial action	Retry the operation after a period of time

In all other circumstances (aside from internal server or SOAP errors) the required Web Service operation is executed.

2.11 hmrcInfo Header

In order to provide HMRC specific metadata about the body of the soap requests an HMRC header is provided. This provides the calling application with the ability to specify information about the software vendor who implemented the client application and also key information about the caller. The HMRC header will be transmitted within the header element of the SOAP envelope. For this release of the NCTS XML Channel service all of the fields in the “hmrcInfo” header will be optional fields.

The following table provides a basic API specification for the proposed HMRCHeader

hmrcInfo		
MessageIdentity	Name	Description
	HMRCMark	This field is reserved for future use and will be ignored by this release of the NCTS XML Channel service <i>Ultimately a SHA-1 hash of the body of the SOAP request to provide one-way non-repudiation to the client..</i>

	ClientMessageID	This field is reserved for future use and will be ignored by this release of the NCTS XML Channel service <i>Ultimately a Client supplied unique message identifier for audit and tracking.</i>	
	HMRCMessageID	This field is reserved for future use and will be ignored by this release of the NCTS XML Channel service <i>Ultimately for End-to-end tracking identifier to support audit, logging and tracking through different systems.</i>	
VendorDetails	<i>Name</i>		
	<i>Description</i>		
	VendorIdentity	VendorID	Unique identifier of the software vendor who implemented the client application used in the invocation of the web service operation. This will be provided during the vendor business registration process.
	VendorIdentity	VendorURI	Vendors Resource Identifier. This will be agreed upon during the vendor business registration process.
	VendorName	Name of the software vendor who provided the client application used in the invocation of the web service operation. This will be agreed upon during the vendor business registration process.	
VendorProduct	VendorProduct	Name of the client application software product used to create the original message and invoke the web service operation.	

		VendorProductVersion	Software version number of the software product, as specified in the <VendorProduct> element, that was to create the original message and invoke the web service operation.
ActionDetails	<i>Name</i>		<i>Description</i>
	ActionKey		This field is reserved for future use and will be ignored by this release of the NCTS XML Channel service <i>Ultimately the service identifier (VRN, TURN etc.) of the party invoking the web service.</i>

Figure 1 Below is a graphical representation of the HMRC header.

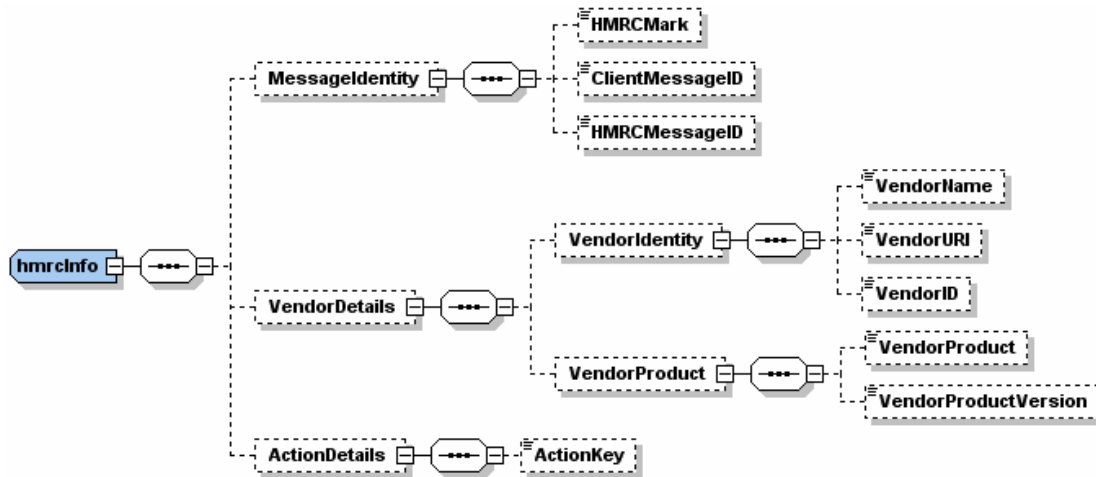


Figure 1 hmrcInfo Header

2.12 Character Encoding

The NCTS Web Service will always use the UTF-8 character encoding when replying to traders' requests. This is in line with the "Simple SOAP Binding Profile 1.0" published by WS-I¹, that clearly states that all SOAP envelopes must be serialized

¹ Footnote: WS-I (Web Services Interoperability Organization) is an open industry organization chartered to promote Web services interoperability across platforms, operating systems, and programming languages. (<http://www.ws-i.org>)

using either UTF-8 or UTF-16 encoding. Clients must also use UTF-8 encoding when invoking the service: it is not guaranteed that requests encoded in other character sets (like ISO-8859-1) will be accepted by the server.

The encoding used in the SOAP exchange will be determined by the value of the "Content-Type" HTTP header, and not by the "encoding" attribute in the XML message header; this is as specified by the "Simple SOAP Binding Profile" mentioned above.

Clients of the web service are responsible for specifying the UTF-8 encoding in the HTTP headers and encoding the message payload accordingly. The steps required to do this will depend on the technologies used to build the clients, and all different alternatives cannot be covered in this document. However, in a modern programming language (like Java or C#), where tools are available to generate client proxies automatically from a provided WSDL file, setting the character encoding is a trivial matter: the caller just needs to set a property in the proxy to set UTF-8 encoding.

3. API Calls

3.1 submitDocument

submitDocument is the Web Service operation invoked to submit an XML message wrapping the IE EDIFACT Request message to be processed. Authentication against the Government Gateway is required for this operation; therefore it is mandatory for the SOAP header to include the WS Security element (see section 2.10 Authentication).

Operation: submitDocument				
Parameters – all Mandatory				
Call	Name	Type	Example Value	Comment
	message	String	<message> UNB+UNOC:..... </message>	The EDIFACT IE request message to be submitted via EDCS to the NCTS system
Return	Name	Type	Example Value	Comment
	header	MessageHeader	<header> </header>	Attributes common to all return types refer to section 3.5.1 MessageHeader
	transactionId	String	0000004453	The unique identifier of a Transaction.

Figure 2 and Figure 3 provide a graphical representation of the types, which represent the interface.

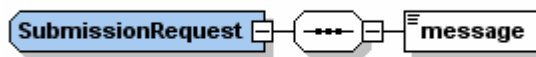


Figure 2 SubmissionRequest

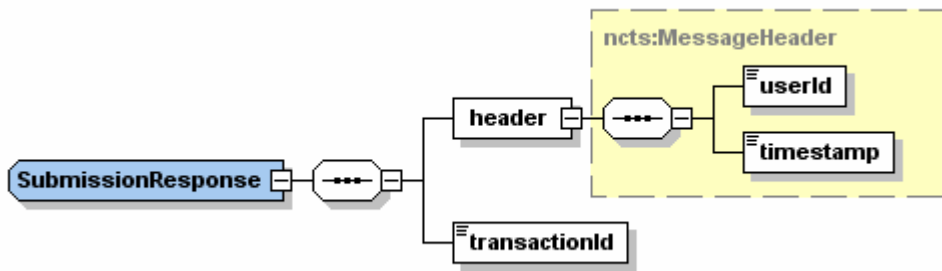


Figure 3 SubmissionResponse

Below are the reported error conditions resulting in the failure of the method call (custom errors raised as SOAP faults)

Bad request	
faultcode	soap:Client.BadArguments
faultString	Missing EDIFACT message.
Cause	The operation was invoked with no IE Request message.
Outcome	No action will be taken other than the generation of this Soap fault
Remedial action	Invoke the operation with an EDIFACT IE message present.

General processing exception	
faultcode	soap:Server
faultString	Internal Server Error
Cause	There was an internal problem performing the operation and it did not complete successfully.
Outcome	The submission of the IE Request message has failed.
Remedial action	Re submit the IE Request message after a period of time

Authentication failures are described in section 3.1 Authentication.

If any technical problems are encountered with the submitDocument call a SOAP fault is returned. The assumption must be, in this case, that the call has failed and in the case of a general processing exception the IE Request must be re-submitted.

3.2 getMessages

getMessages is the Web Service operation invoked to request the response(s) to previous Submission Requests received by the system for the identified Trader with a Response Sequence Number (strictly) greater than the one specified. Authentication against the Government Gateway is required for this operation; therefore it is mandatory for the SOAP header to include the WS Security element (see section 2.10 Authentication).

Operation: getMessages				
Parameters — maxResponses Optional				
Call	Name	Type	Example Value	Comment
	lastRetrieved	Integer	100	Response sequence number of the highest message previously retrieved. It is this value+1 which will indicate the point in the sequence of response where retrieval will begin.
	maxResponses	Integer	10	The maximum number of IE Response messages to be retrieved.
Return	Name	Type	Example Value	Comment
	header	MessageHeader	<header> </header>	Attributes common to all return types refer to section 3.5.1 MessageHeader
	HighestReturned	String	110	The message sequence ID of the highest message returned. Only present if there are one or more messages returned
	MoreAvailable	boolean	false	Flag set to indicate whether or not there are more Response messages available than have been returned. Only present if there are messages returned
	messages	ResponseMessage []	<messages> </messages>	An array of response messages Refer to section 3.5.2 ResponseMessage

Figure 4 and Figure 5 provide a graphical representation of the types, which represent the interface.

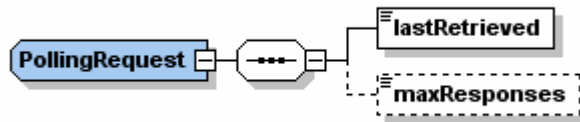


Figure 4 PollingRequest

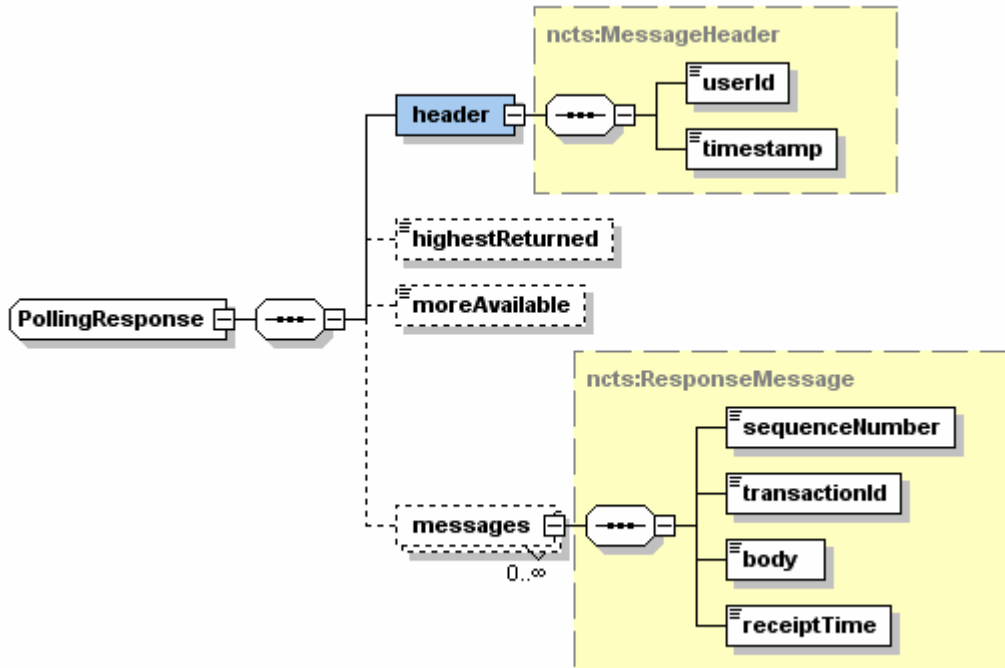


Figure 5 PollingResponse

Below are the reported error conditions resulting in the failure of the method call (custom errors raised as SOAP faults).

Bad request	
faultcode	soap:Client.BadArguments
faultString	Missing Sequence Number parameter.
Cause	The operation was invoked with no Sequence Number supplied.
Outcome	No action will be taken other than the generation of this Soap fault
Remedial action	Invoke the operation with a Sequence Number present

Bad request	
faultcode	soap:Client.BadArguments
faultString	The provided sequence number [<provided sequence number>] was invalid. Retry with a number between [0] and [<lastAvailableResponse>].
Cause	The sequence number provided in the “lastRetrieved” element was not valid, in that it did not belong to an existing IE Response message.
Outcome	No action will be taken other than the generation of this Soap fault
Remedial action	Invoke the operation with a sequence number that is within the specified valid range

General processing exception	
faultcode	soap:Server
faultString	Internal Server Error
Cause	There was an internal problem performing the operation and it did not complete successfully.
Outcome	The getMessage() operation has failed.
Remedial action	Rerty after a period of time

Where a “getMessages” legitimately results in no IE Response messages being identified for retrieval, the SOAP response will contain no <messages> elements and no <highestReturned> or <moreAvailable>.

3.3 GetMessagesByTransactionId

getMessagesByTransactionId is the Web Service operation invoked to request the response(s) to a given transaction identified by the specified Transaction Identifier. Authentication against the Government Gateway is required for this operation; therefore it is mandatory for the SOAP header to include a WS Security element (see section 2.10 Authentication).

Operation: getMessages				
Parameters – all Mandatory				
Call	Name	Type	Example Value	Comment
	transactionId	String	0000004453	The unique identifier of a Transaction who's associated responses will be returned.
Return	Name	Type	Example Value	Comment
	header	MessageHeader	<header> </header>	Attributes common to all return types refer to section 3.5.1 MessageHeader
	messages	ResponseMessage []	<messages> </messages>	An array of response messages refer to section 3.5.2 ResponseMessage

Figure 6 and Figure 7 provide a graphical representation of the types, which represent the interface.



Figure 6 GetByTransactionIDRequest

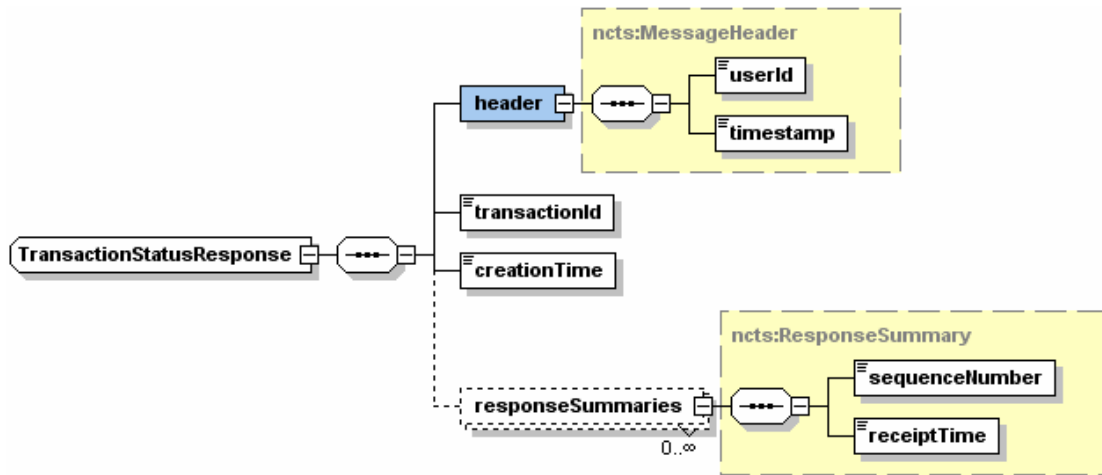


Figure 7 GetByTransactionResponse

Below are the reported error conditions resulting in failure of method call (custom errors raised as SOAP faults)

Bad request	
faultcode	soap:Client.BadArguments
faultString	Missing Transaction Id parameter.
Cause	The operation was invoked with no Transaction ID supplied.
Outcome	No action will be taken other than the generation of this Soap fault
Remedial action	Invoke the operation with an Transaction ID present

Bad request	
faultcode	soap:Client.BadArguments
faultString	The requested Transaction Id [<transactionId>] does not exist
Cause	The Transaction ID provided in the “transactionid” element was not valid, in that it did not belong to an existing transaction associated with this user.
Outcome	No action will be taken other than the generation of this Soap fault
Remedial action	Invoke the operation with an existing valid Transaction ID.

General processing exception	
faultcode	soap:Server
faultString	Internal Server Error
Cause	There was an internal problem performing the operation and it did not complete successfully.
Outcome	The getMessagesByTransactionId operation has failed.
Remedial action	Retry after a period of time

Where a “getMessagesByTransactionId” call legitimately results in no IE Response messages being identified for retrieval, the SOAP response will contain no <messages> elements.

3.4 getTransactionStatus

getTransactionStatus is the Web Service operation invoked to request the status of a specific Transaction. Authentication against the Government Gateway is required for this operation; therefore it is mandatory for the SOAP header to include a WS Security element (see section 2.10 Authentication).

Operation: getMessages				
Parameters – all Mandatory				
Call	Name	Type	Example Value	Comment
	transactionId	String	0000004453	The unique identifier of the Transaction.
Return	Name	Type	Example Value	Comment
	header	MessageHeader	<header> </header>	Attributes common to all return types, refer to section 3.5.1 MessageHeader
	TransactionId	String	0000004453	The unique identifier of the Transaction.
	CreationTime	Date	2006-01-18T15:58:15.984	The time, to the millisecond that the IE request message was persisted in the XML channel data store.
	ResponseSummaries	ResponseSummary []	<responseSummaries> ... </responseSummaries>	Array of ResponseSummary objects

Figure 8 and Figure 9 provide a graphical representation of the types, which represent the interface.

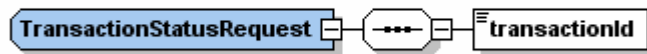


Figure 8 TransactionStatusRequest

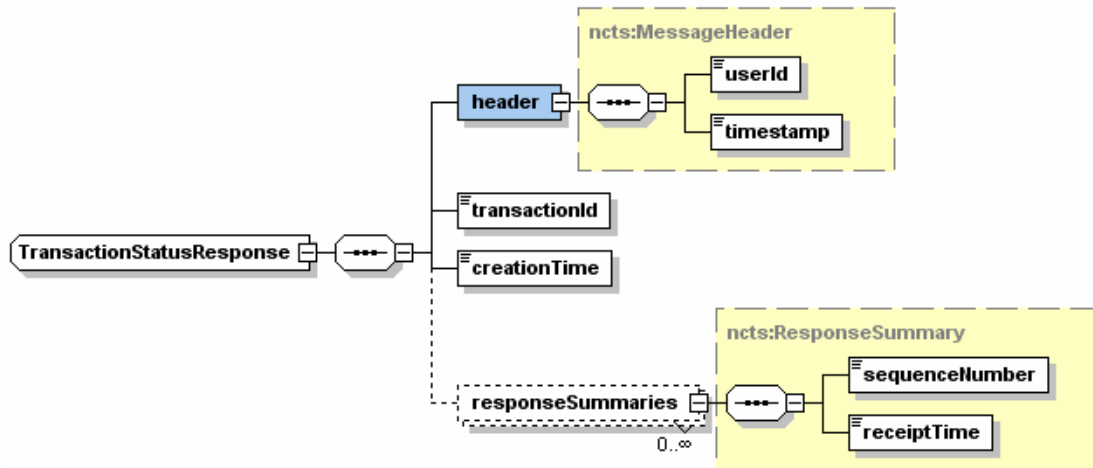


Figure 9 TransactionStatusResponse

Below are the reported error conditions resulting in the failure of the method call (custom errors raised as SOAP faults)

Bad request	
faultcode	soap:Client.BadArguments
faultString	Missing Transaction Status request parameter.
Cause	The operation was invoked with no Transaction ID supplied.
Outcome	No action will be taken other than the generation of this Soap fault
Remedial action	Invoke the operation with a Transaction ID present

Bad request	
faultcode	soap:Client.BadArguments
faultString	The requested Transaction Id [<transactionId>] does not exist
Cause	The Transaction ID provided in the “transactionid” element was not valid, in that it did not belong to an existing transaction associated with this user.
Outcome	No action will be taken other than the generation of this Soap fault
Remedial action	Invoke the operation with an existing valid Transaction ID.

General processing exception	
faultcode	soap:Server
faultString	Internal Server Error
Cause	There was an internal problem performing the operation and it did not complete successfully.
Outcome	The getMessagesByTransactionId operation has failed.
Remedial action	Retry after a period of time

When a “getTransactionStatus” call results in no IE Response messages being identified as being associated with the IE Request specified the SOAP response would contain no <responseSummaries> elements, this is not an error conditions.

3.5 Supporting types

3.5.1 MessageHeader

Holder for values common to all SOAP Responses.

Element - ncts:MessageHeader			
<i>Name</i>	<i>Type</i>	<i>Example Value</i>	<i>Comment</i>
userId	String	testuser1	The Gateway userID that was used to authenticate the Web Service call
timestamp	Date	2006-01-18T15:58:17.031	Time stamp indicating the execution time of the operation.

3.5.2 ResponseMessage

All information related to a specific IE Response message.

Element - ncts:ResponseMessage			
<i>Name</i>	<i>Type</i>	<i>Example Value</i>	<i>Comment</i>
sequenceNumber	Integer	100	Every IE Response message is tagged with a progressively increasing sequence number specific to each Trader

transactionId	String	0000004453	The unique identifier of the Transaction. The transaction ID is optional and if absent indicates that this response is not correlated with an originating request
body	String	UNB+UNOC:3....	The IE EDIFACT Response message
receiptTime	Date	2006-01-18T15:58:17.031	The date/time that the response was stored in the NCTS XML Channel data store.

3.5.3 ResponseSummary

Succinct information regarding an IE Response message received for a given Submission Request.

Element - ncts:ResponseSummary			
<i>Name</i>	<i>Type</i>	<i>Example Value</i>	<i>Comment</i>
sequenceNumber	Integer	100	Every IE Response message is tagged with a progressively increasing sequence number specific to each Trader
receiptTime	Date	2006-01-18T15:58:17.031	The date/time that the response was stored

4. Appendix A – Example SOAP Messages

The following XML fragments represent examples of the input and output of each of the Web Service operations that are the subject of this API document. For clarity the standard XML headers and name space values have been omitted from the fragments.

Each SOAP element tag is preceded by a SOAP namespace prefix, for example <env:Header>. The schema defined by each of these is:

env: indicates that the schema used is the SOAP Envelope schema

wsse: indicates that the schema used is the OASIS Web Service Security schema

ncts: indicates that the schema used is the HMRC NCTS schema

4.1 submitDocument

4.1.1 SOAP Request

```
<env:Envelope>
  <env:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>Trader1</wsse:Username>
        <wsse:Password>testing1</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
    <hmrc:Header>
      <hmrc:VendorDetails>
        <hmrc:Vendorname>Vender1</ hmrc:Vendorname>
        <hmrc:VendorURI>Vender1</ hmrc:VendorURI>
        <hmrc:VendorID>V10000</ hmrc:VendorID>
        <hmrc:VendoProduct>Client Product 1</ hmrc:VendorProduct>
        <hmrc:VendorOroductVersion>1.0</ hmrc:VendorProductVersion>
      </hmrc:VendorDetails>
      <hmrc:ActionDetails>
        <hmrc:ActionKey>123456789123</ hmrc:ActionKey>
      </hmrc:ActionDetails>
    </hmrc:Header>
  </env:Header>
  <env:Body>
    <ncts:submitDocument>
      <ncts:message>
        UNB+UNOC:3+0000123456789+NCTS+050925:1041+GR050925104106++NCTS'.....
      </ncts:message>
    </ncts:submitDocument>
  </env:Body>
</env:Envelope>
```

4.1.2 SOAP Response

```
<env:Envelope>
  <env:Body>
    <ncts:submitDocumentResponse>
      <ncts:header>
        <ncts:userId>Trader1</ncts:userId>
        <ncts:timestamp>2006-01-18T15:58:16.078Z</ncts:timestamp>
      </ncts:header>
      <ncts:transactionId>7599613860</ncts:transactionId>
    </ncts:submitDocumentResponse>
  </env:Body>
</env:Envelope>
```

4.2 getMessages

4.2.1 SOAP Request

```
<env:Envelope>
  <env:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>Trader1</wsse:Username>
        <wsse:Password>testing1</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
    <hmrc:Header>
      <hmrc:VendorDetails>
        <hmrc:Vendorname>Vender1</ hmrc:Vendorname>
        <hmrc:VendorURI>Vender1</ hmrc:VendorURI>
        <hmrc:VendorID>V10000</ hmrc:VendorID>
        <hmrc:VendoProduct>Client Product 1</ hmrc:VendorProduct>
        <hmrc:VendorOroductVersion>1.0</ hmrc:VendorProductVersion>
      </hmrc:VendorDetails>
      <hmrc:ActionDetails>
        <hmrc:ActionKey>123456789123</ hmrc:ActionKey>
      </hmrc:ActionDetails>
    </hmrc:Header>
  </env:Header>
  <env:Body>
    <ncts:getMessages>
      <ncts:lastRetrieved >100</ncts:lastRetrieved>
      <ncts:maxResponses>100</ncts:maxResponses>
    </ncts:getMessages>
  </env:Body>
</env:Envelope>
```

4.2.2 SOAP Response

```
<env:Envelope>
  <env:Body>
    <ncts:getMessagesResponse>
      <ncts:header>
        <ncts:userId>Trader1</ncts:userId>
        <ncts:timestamp>2006-01-18T15:58:18.140Z</ncts:timestamp>
      </ncts:header>
      <ncts:highestReturned>102</ncts:highestReturned>
      <ncts:moreAvailable>>false</ncts:moreAvailable>
      <ncts:messages>
        <ncts:sequenceNumber>101</ncts:sequenceNumber>
        <ncts:transactionId>7599613860</ncts:transactionId>
        <ncts:body>
          UNB+UNOC:3+NTA.GB+ABC123456+060117:0906+95060117090651++NCTS.....
        </ncts:body>
        <ncts:receiptTime>2006-01-18T15:58:17.031Z</ncts:receiptTime>
      </ncts:messages>
      <ncts:messages>
        <ncts:sequenceNumber>102</ncts:sequenceNumber>
        <ncts:transactionId>7599613860</ncts:transactionId>
        <ncts:body>
          UNB+UNOC:3+NTA.GB+ABC123456+060117:0906+95060117090651++NCTS.....
        </ncts:body>
        <ncts:receiptTime>2006-01-18T15:58:20.031Z</ncts:receiptTime>
      </ncts:messages>
    </ncts:getMessagesResponse>
  </env:Body>
</env:Envelope>
```

4.3 *getMessagesByTransactionId*

4.3.1 SOAP Request

```
<env:Envelope>
  <env:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>Trader1</wsse:Username>
        <wsse:Password>testing1</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
    <hmrc:Header>
      <hmrc:VendorDetails>
        <hmrc:Vendorname>Vender1</ hmrc:Vendorname>
        <hmrc:VendorURI>Vender1</ hmrc:VendorURI>
        <hmrc:VendorID>V10000</ hmrc:VendorID>
        <hmrc:VendoProduct>Client Product 1</ hmrc:VendorProduct>
        <hmrc:VendorOroductVersion>1.0</ hmrc:VendorProductVersion>
      </hmrc:VendorDetails>
      <hmrc:ActionDetails>
        <hmrc:ActionKey>123456789123</ hmrc:ActionKey>
      </hmrc:ActionDetails>
    </hmrc:Header>
  </env:Header>
  <env:Body>
    <ncts:getMessagesByTransactionId>
      <ncts:transactionId>7599613860<ncts:transactionId>
    </ncts:getMessagesByTransactionId>
  </env:Body>
</env:Envelope>
```

4.3.2 SOAP Response

```
<env:Envelope>
  <env:Body>
    <ncts:getMessagesByTransactionIdResponse>
      <ncts:header>
        <ncts:userId>Trader1</ncts:userId>
        <ncts:timestamp>2006-01-18T15:58:18.140Z</ncts:timestamp>
      </ncts:header>
      <ncts:messages>
        <ncts:sequenceNumber>101</ncts:sequenceNumber>
        <ncts:transactionId>7599613860</ncts:transactionId>
        <ncts:body>
          UNB+UNOC:3+NTA.GB+ABC123456+060117:0906+95060117090651++NCTS.....
        </ncts:body>
        <ncts:receiptTime>2006-01-18T15:58:17.031Z</ncts:receiptTime>
      </ncts:messages>
      <ncts:messages>
        <ncts:sequenceNumber>102</ncts:sequenceNumber>
        <ncts:transactionId>7599613860</ncts:transactionId>
        <ncts:body>
          UNB+UNOC:3+NTA.GB+ABC123456+060117:0906+95060117090651++NCTS.....
        </ncts:body>
        <ncts:receiptTime>2006-01-18T15:58:20.031Z</ncts:receiptTime>
      </ncts:messages>
    </ncts:getMessagesByTransactionIdResponse>
  </env:Body>
</env:Envelope>
```

4.4 getTransactionStatus

4.4.1 SOAP Request

```
<env:Envelope>
  <env:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>Trader1</wsse:Username>
        <wsse:Password>testing1</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </env:Header>
  <hmrc:Header>
    <hmrc:VendorDetails>
      <hmrc:Vendorname>Vender1</ hmrc:Vendorname>
      <hmrc:VendorURI>Vender1</ hmrc:VendorURI>
      <hmrc:VendorID>V10000</ hmrc:VendorID>
      <hmrc:VendoProduct>Client Product 1</ hmrc:VendorProduct>
      <hmrc:VendorOproductVersion>1.0</ hmrc:VendorProductVersion>
    </hmrc:VendorDetails>
    <hmrc:ActionDetails>
      <hmrc:ActionKey>123456789123</ hmrc:ActionKey>
    </hmrc:ActionDetails>
  </hmrc:Header>
  <env:Body>
    <ncts:getTransactionStatus>
      <ncts:transactionId>7599613860</ncts:transactionId>
    </ncts:getTransactionStatus>
  </env:Body>
</env:Envelope>
```

4.4.2 SOAP Response

```
<env:Envelope>
  <env:Body>
    <ncts:getTransactionStatusResponse>
      <ncts:header>
        <userId>Trader1</userId>
        <ncts:timestamp>2006-01-18T15:58:18.328Z</ncts:timestamp>
      </ncts:header>
      <ncts:transactionId>7599613860</ncts:transactionId>
      <ncts:creationTime>2006-01-18T15:58:15.984Z</ncts:creationTime>
      <ncts:responseSummaries>
        <ncts:sequenceNumber>101</ncts:sequenceNumber>
        <ncts:receiptTime>2006-01-18T15:58:17.031Z</ncts:receiptTime>
      </ncts:responseSummaries>
      <ncts:responseSummaries>
        <ncts:sequenceNumber>102</ncts:sequenceNumber>
        <ncts:receiptTime>2006-01-18T15:58:20.031Z</ncts:receiptTime>
      </ncts:responseSummaries>
    </ncts:getTransactionStatusResponse>
  </env:Body>
</env:Envelope>
```

5. Appendix B SOAP Fault Examples

5.1 *SubmitDocument*

5.1.1 Bad request

```
<env:Envelope>  
  <env:Body>  
    <env:Fault>  
      <faultcode>  
        soap:Client.BadArguments  
      </faultcode>  
      <faultstring>  
        The operation was invoked with no IE Request message.  
      </faultstring>  
    </env:Fault>  
  </env:Body>  
</env:Envelope>
```

5.1.2 General processing exception

```
<env:Envelope>  
  <env:Body>  
    <env:Fault>  
      <faultcode>  
        soap:Server  
      </faultcode>  
      <faultstring>  
        Internal Server Error  
      </faultstring>  
    </env:Fault>  
  </env:Body>  
</env:Envelope>
```

5.2 *getMessages*

5.2.1 Bad request

```
<env:Envelope>
  <env:Body>
    <env:Fault>
      <faultcode>
        fault:Client.BadArguments
      </faultcode>
      <faultstring>
        Missing Sequence Number parameter.
      </faultstring>
      <detail/>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

```
<env:Envelope>
  <env:Body>
    <env:Fault>
      <faultcode>
        fault:Client.BadArguments
      </faultcode>
      <faultstring>
        The provided sequence number [<provided sequence number>] was invalid. Retry with a number
        between [0] and [<lastAvailableResponse>].
      </faultstring>
      <detail/>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

5.2.2 General processing exception

```
<env:Envelope>
  <env:Body>
    <env:Fault>
      <faultcode>
        soap:Server
      </faultcode>
      <faultstring>
        Internal Server Error
      </faultstring>
      <detail/>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

5.3 *getMessagesByTransactionId*

5.3.1 Bad request

```
<env:Envelope>  
  <env:Body>  
    <env:Fault>  
      <faultcode>  
        fault:Client.BadArguments  
      </faultcode>  
      <faultstring>  
        Missing Transaction Id parameter.  
      </faultstring>  
    </env:Fault>  
  </env:Body>  
</env:Envelope>
```

```
<env:Envelope>  
  <env:Body>  
    <env:Fault>  
      <faultcode>  
        fault:Client.BadArguments  
      </faultcode>  
      <faultstring>  
        The requested Transaction Id [<transactionId>] does not exist  
      </faultstring>  
    </env:Fault>  
  </env:Body>  
</env:Envelope>
```

5.3.2 General processing exception

```
<env:Envelope>  
  <env:Body>  
    <env:Fault>  
      <faultcode>  
        soap:Server  
      </faultcode>  
      <faultstring>  
        Internal Server Error  
      </faultstring>  
    </env:Fault>  
  </env:Body>  
</env:Envelope>
```

5.4 GetTransactionStatus

5.4.1 Bad request

```
<env:Envelope>
  <env:Body>
    <env:Fault>
      <faultcode>
        fault:Client.BadArguments
      </faultcode>
      <faultstring>
        Missing Transaction Status request parameter.
      </faultstring>
      <detail/>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

```
<env:Envelope>
  <env:Body>
    <env:Fault>
      <faultcode>
        fault:Client.BadArguments
      </faultcode>
      <faultstring>
        The requested Transaction Id [<transactionId>] does not exist
      </faultstring>
      <detail/>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

5.4.2 General processing exception

```
<env:Envelope>
  <env:Body>
    <env:Fault>
      <faultcode>
        soap:Server
      </faultcode>
      <faultstring>
        Internal Server Error
      </faultstring>
      <detail/>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

6. Appendix B Notes on EDIFACT encoding

The EDIFACT standard predates the widespread use of Unicode and UTF encodings. As a result, the resulting standard can be confusing for those trying to send messages including non-Western characters:

An NCTS EDIFACT message can use characters from the ISO-8859-1 (Western European) set only. However, "language fields" are defined in the message payload through which certain fields, like the destination address, can be set to be in a different language (like Greek), in which case the characters in that field (and only that field) will be interpreted by the backend as if belonging to a different set (ISO-8859-7, Greek, in this case). This way of combining multiple character sets into one final ISO-8859-1 message is independent of the channel through which the message is transmitted later on.

In our case, a message containing ISO-8859-1 characters (i.e. an EDIFACT message) is going to be transmitted over an UTF-8 HTTP channel. Those characters in the lower half of the table (positions 1-128, including all western alphanumerical characters and punctuation marks) are transmitted in UTF-8 using only one byte; those in the upper half of the table (like 'á' or 'ü') will be encoded in UTF-8 using two bytes.

Software developers should be aware of the fact that one character can require more than one byte; in some old languages like C, where the one-char-one-byte assumption is widespread, developers need to be particularly careful, as conversion routines might be needed. Java and C#, among other languages, treat all strings internally as Unicode; when the string is written onto a stream of bytes (like a network socket), all the necessary conversions are handled transparently by the language.