Document filename: ITK 2 0 Web Service Transport Requirements v1.0.docx			
Directorate / Programme :	HSCIC - Architecture	**Project**	Interoperability
Document Reference :		HSCIC-ITK-ARCH-104	
Project Manager :	Rob Shaw	**Status :**	Final
Owner :	George Hope	**Document Version :**	1.1
Author :	George Hope	**Version issue date :**	14/07/2014

ITK Web Service Transport Requirements

Document Management

Revision History

Version	Date	Summary of Changes
1.0	31/05/2014	First version issued by HSCIC
1.1	14/07/2014	Included SOAP 1.2 fault handling (see section 6 for details)

Reviewers

This document was reviewed by the following people:

Reviewer name	Title / Responsibility	Date	Version
George Hope	ITK Architecture Lead	30/04/2014	1.0
Sanjay Paul	ITK Architect	30/04/2014	1.0
Richard Dobson	ITK Accreditation Manager	30/04/2014	1.0
David Barnet	ITK Communication and Messaging	30/04/2014	1.0
Nigel Saville	ITK Accreditation	30/04/2014	1.0

Approved by

This document was approved by the following people:

Name	Signature	Title	Date	Version
Shaun Fletcher		Head of Architecture	31/05/2014	1.0
Rob Shaw		Director Operational Services	31/05/2014	1.0

Reference Documents

Ref no	Doc Reference Number	Title	Version
1.			
2.			
3.			
4.			

Document Control:

The controlled copy of this document is maintained in the HSCIC corporate network. Any copies of this document held outside of that area, in whatever format (e.g. paper, email attachment), are considered to have passed out of control and should be checked for currency and validity.

Contents

1 Introduction

This document forms part of the overall document set for ITK Architecture.

1.1 Purpose of Document

This document defines a set of requirements for ITK Web Services Transport.

1.2 ITK Architecture Documentation Set

The position of this document in relation to the document set is shown below.

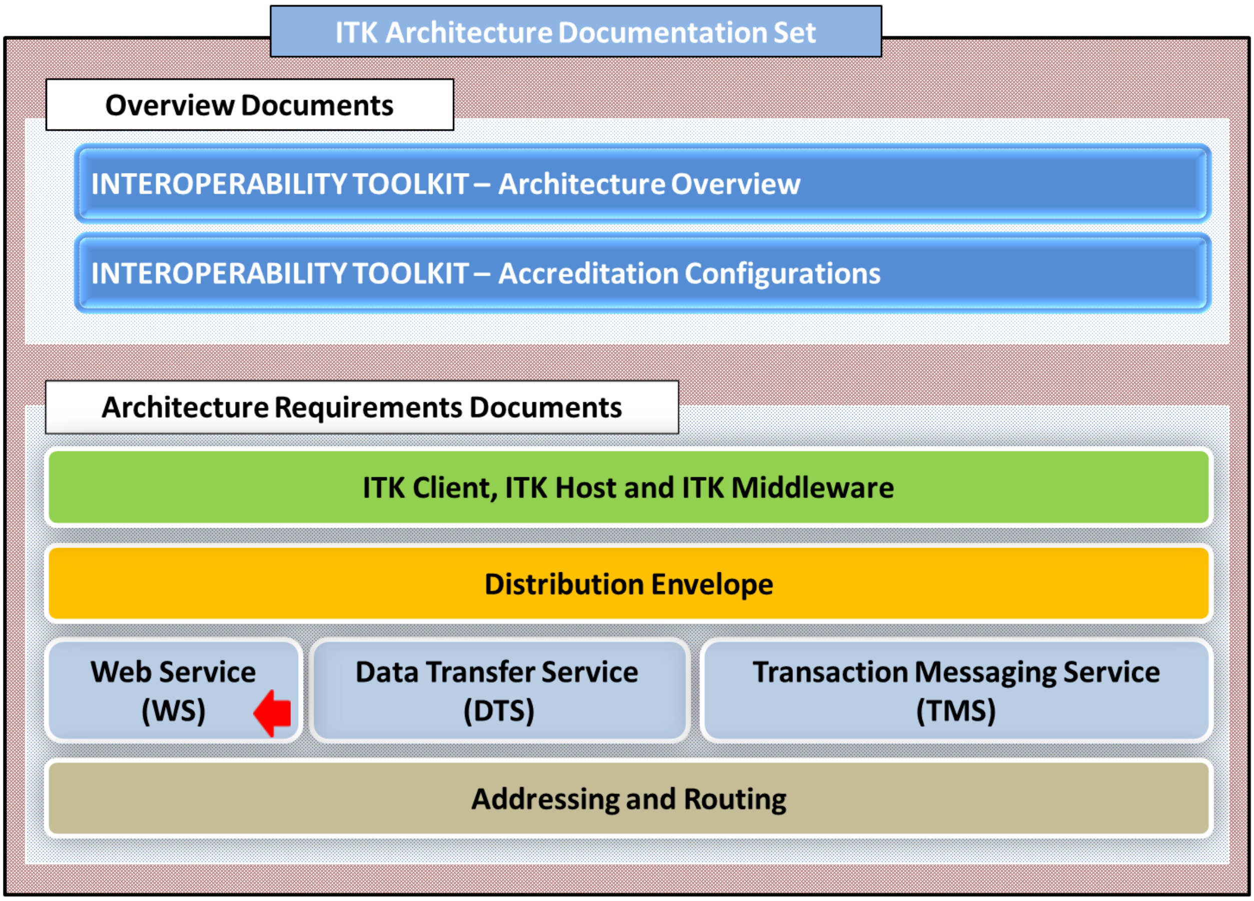

ITK Architecture Documentation Set
Overview Documents
INTEROPERABILITY TOOLKIT – Architecture Overview
INTEROPERABILITY TOOLKIT – Accreditation Configurations
Architecture Requirements Documents
ITK Client, ITK Host and ITK Middleware
Distribution Envelope
Web Service (WS)
Data Transfer Service (DTS)
Transaction Messaging Service (TMS)
Addressing and Routing

Figure 1 – ITK Architecture Documentation Set

1.3 Audience

The primary audience are supplier technical and product development staff who are interested in developing a Toolkit Implementation.

1.4 Document Scope

The document covers the ITK Web Services Transport Interoperability requirements required for accreditation.

1.5 Document Overview

The rest of this document covers a number of areas of functionality. Within each area the functionality is described, and a number of formal requirements are listed in bold type, with additional detail provided in smaller type below this.

1.6 Requirements Presentation

The requirements are presented in the format given below:

Ref (1)	**Description (2)**	**Client (3)**	**Host (4)**	**MW (5)**	**SMSP (6)**
COR-REL-03	**Toolkit Implementations MUST retain responsibility for processing until a request completes**	**Y**	**N**	**Y**	**N**
NB (7)	Specifically, any response returned from the initial part of the asynchronous invocation does NOT indicate a transfer of responsibility. It is only a transport acknowledgement, and it does NOT imply that the message has necessarily been persisted, nor does it indicate a transfer of responsibility, nor promise that subsequent application processing will be completed.				

Clarification Notes

(1) The requirement reference

(2) The Description of the requirement

(3), (4), (5) and **(6)** Shows the requirements applicability for accreditation

(7) Provides further details relating to the requirement and supplementary notes

Colour Coding Notes

- The fill colour of the Reference relates to a particular document from the document map.
- Where requirements are universally applied the fill colour will always be blue. Where requirements are conditional and may impact accreditation the fill colour will be Orange.
- See the Accreditation Configuration spread sheet for related details.

1.7 Reference Implementation

An ITK reference implementation pack is available as a training and development aid and it contains example code snippets for typical Healthcare Interoperability scenarios.

http://developer.nhs.uk/library/interoperability/nhs-interoperability-framework/

2 Web Service Overview

2.1 Message Structure Overview

The diagram below shows an overview of an ITK Web Service message.

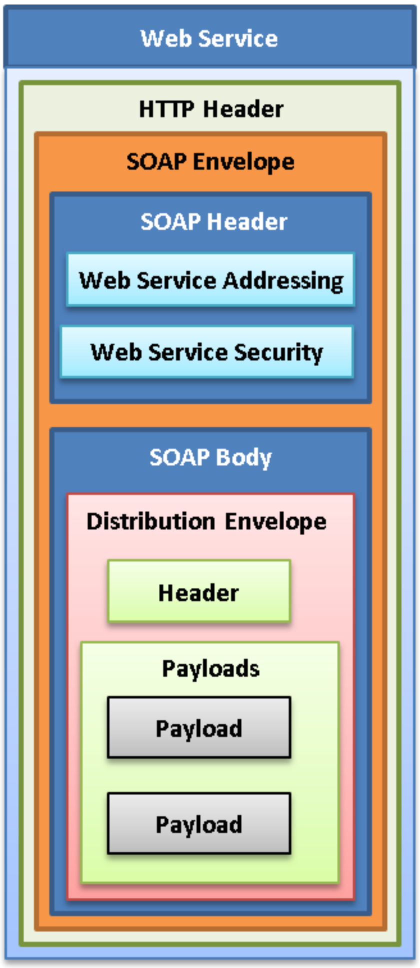
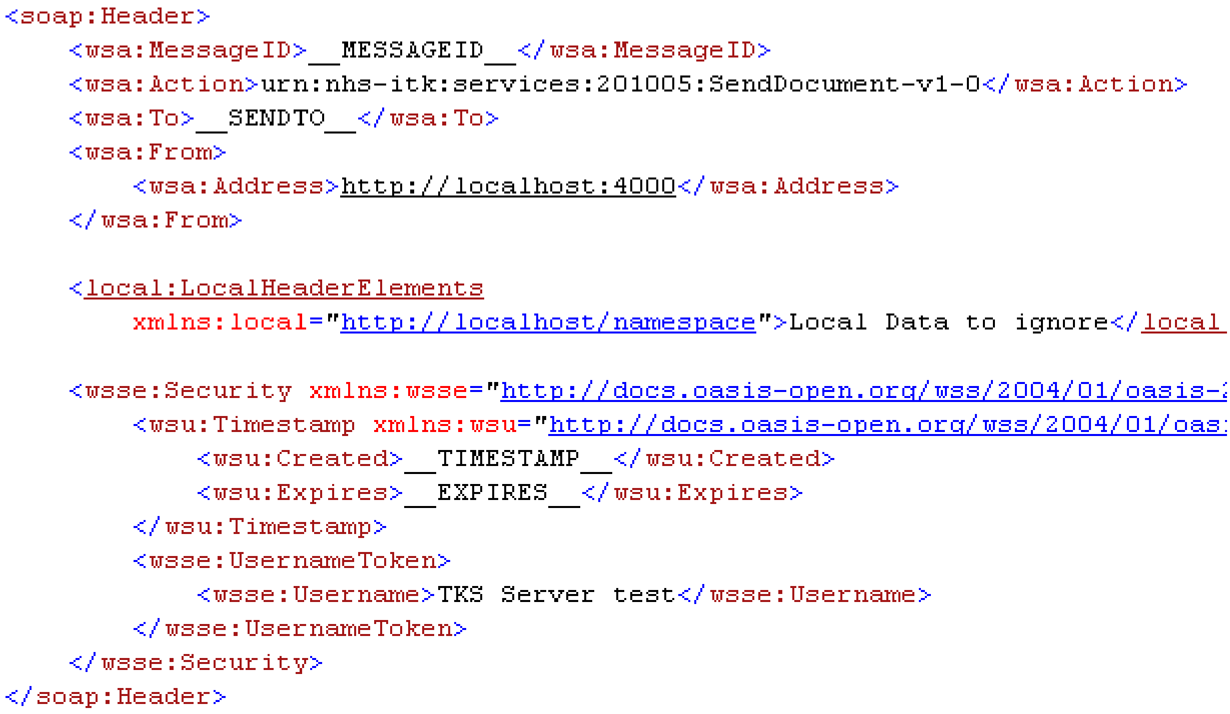


Figure 2 - An ITK Web Service Message Structure

2.2 SOAP Header

The SOAP Header provides generic functionality common to all messages and relating to technical distribution over a single messaging “hop”. There are a defined set of ITK related elements to enable healthcare interoperability.



```
<soap:Header>
    <wsa:MessageID>__MESSAGEID__</wsa:MessageID>
    <wsa:Action>urn:nhs-itk:services:201005:SendDocument-v1-0</wsa:Action>
    <wsa:To>__SENDTO__</wsa:To>
    <wsa:From>
        <wsa:Address>http://localhost:4000</wsa:Address>
    </wsa:From>

    <local:LocalHeaderElements
        xmlns:local="http://localhost/namespace">Local Data to ignore</local

    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
        <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oas
            <wsu:Created>__TIMESTAMP__</wsu:Created>
            <wsu:Expires>__EXPIRES__</wsu:Expires>
        </wsu:Timestamp>
        <wsse:UsernameToken>
            <wsse:Username>TKS Server test</wsse:Username>
        </wsse:UsernameToken>
    </wsse:Security>
</soap:Header>
```

SOAP 1.1 used. Known ITK SOAP Elements are

WS Addressing:

- \MessageID
- \To
- \Action
- \From
- \ReplyTo
- \FaultTo
- \RelatesTo

WS Security:

- \UsernameToken
- \UsernameToken\Created
- \UsernameToken\Expires

Any other SOAP Elements must be within a local namespace. Known ITK SOAP Elements can only have attribute mustUnderstand=“1”. The following sections explain the ITK approach in detail.

2.2.1 SOAP Header Explained

The table below provides the detailed requirements associated with the ITK Soap Header.

Requirement	SOAP Header	Cardinality	Details
WS-ADR-01	\MessageID	1..1	Must be unique and in upper case.
WS-ADR-03	\To	1..1	Contains the web service endpoint
WS-ADR-04	\Action	1..1	The SOAP action binding is detailed within the appropriate domain message specification, specifically within a wsdl file. In this worked example (urn:nhs-itk:services:201005:createPatient-v1-0) the wsdl is sourced from the TRUD hosted ‘NHS Interoperability Toolkit HL7v2’ domain message specification.
WS-ADR-05	\From	0..1	May be omitted, listening web service should not require this field to be populated.
WS-ADR-06	\ReplyTo	0..1	Must be populated for asynchronous service calls. Responding system will use this address for asynchronous SOAP response.
WS-ADR-06	\FaultTo	0..1	MAY be populated for asynchronous service calls. Responding system will use this address for asynchronous SOAP fault response.
WS-ADR-07	\RelatesTo	0..1	Must be populated for asynchronous SOAP responses, containing the originating SOAP request MessageID.
WS-SEC-02	\UsernameToken	1..1	This should contain the calling systems identifier. The UsernameToken should be the same as the Subject field CN within the presented TLS certificate. .
WS-SEC-04	\Created \Expires	1..1	All header timestamps MUST be in GMT/UTC and MUST be synchronised with a consistent time source to within 250 milliseconds.

Table 1 : SOAP Header Element Definitions

2.2.2 SOAP Header Post Processing

The table below provides an overview the element definitions of the SOAP Header Elements post processing.

Requirement	SOAP Header	Cardinality	Host Processing Details
WS-ADR-01	\MessageID	1..1	Reject with a SOAP fault message if this field is missing.
WS-ADR-03	\To	1..1	Reject with a SOAP fault message if this field is missing. If the address cannot be resolved, reject with a SOAP Fault.
WS-ADR-04	\Action	1..1	Reject with a SOAP fault if the service does not exist.
WS-ADR-05	\From	0..1	None
WS-ADR-06	\ReplyTo	0..1	IF using Synchronous web service and this header is included, then the host system needs to check it is using the Anonymous address - http://www.w3.org/2005/08/addressing/anonymous If not the anonymous address then respond with a SOAP fault. Asynchronous service call – host system will us this address for the SOAP response.
WS-ADR-06	\FaultTo	0..1	Can be populated and if present this is the address used by the host for asynchronous SOAP faults.
WS-ADR-07	\RelatesTo	0..1	Will only be present if the message is an asynchronous response.
WS-SEC-02	\UsernameToken	1..1	The host should check the username is trusted. Host has the option to additionally check of the username matched the TLS MA presented certificates subject field CN= value.
WS-SEC-04	\Created \Expires	1..1	Host needs to check the message has not expired and that the created time is not after the expired time.

Table 2 : SOAP Header Post Processing

2.3 SOAP Body – Distribution Envelope

For all services, the SOAP Body contains an ITK “Distribution Envelope”. This carries information relating to the end-to-end technical distribution of a message and encloses the clinical Payload.

Detailed requirements for the ITK Distribution Envelope can be found in the Distribution Envelope Requirements Specification.

3 Web Service Standards

The following are the Web Service standards used by the ITK.

Standard	Notes
XML [1.0] and XML Schema [1.1]	As per standards defined by W3C.
WS-I Basic Profile [1.1]	This standard restricts and clarifies the usage of other specifications
HTTP [1.1]	With restrictions and clarifications as per the WS-Basic Profile
SOAP [1.1]	http://schemas.xmlsoap.org/soap/envelope With restrictions and clarifications as per the WS-Basic Profile
WSDL [1.1]	With restrictions and clarifications as per the WS-Basic Profile
WS-Addressing [1.0]	http://www.w3.org/2005/08/addressing
WS-Security [1.1]	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
SAML [1.1]	https://www.oasis-open.org/standards#samlv1.1
XML Signature [1.1]	http://www.w3.org/TR/xmldsig-core/
XML Encryption [1.1]	http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
UDDI [3.0]	https://www.oasis-open.org/standards#uddiv3
ITK [2.0] urn:nhs-itk:ns:201005	Additional items explicitly defined as part of the Interoperability Toolkit.
UTF-8	All Toolkit Messages MUST use UTF-8 encoding.

Table 3 : List of Standards

3.1 ITK Web Service Standards

The following are the ITK Requirements associated with the Web Service standards.

Ref	Description	Client	Host	MW	SMSP
WS-STD-01	**Standards used for ITK Web Services Implementations MUST be selected from those listed in the table above**	Y	Y	Y	Y
NB	The standards in the table above have been selected as mature and widely accepted Web Services standards.				

WS-STD-03	**SOAP Header elements MUST be nested directly underneath a single SOAP “Header” tag**	Y	Y	Y	Y
NB	For the purposes of the Toolkit, all of the SOAP Header elements described in this specification MUST be nested directly under a single SOAP “Header.				

WS-STD-04	**xsi:schemaLocation attribute MUST NOT be used**	Y	Y	Y	Y

WS-PAT-04	**The SimpleMessageResponse element of the SOAP Body MUST contain a simple acknowledgement**	N	N	N	Y
1	The payload element of this SimpleMessageResponse MUST contain a simple “OK” string. It MUST NOT contain any significant information which the recipient is expected to understand or process.				

WS-PAT-05	**The appropriate pattern and invocation style MUST be used for each ITK defined service.**	Y	Y	Y	Y
NB	Details can be found in the Accreditation Pack - Service Listings, for the Message Configuration / invocations that must be used.				

3.2 Reliability

Ref	Description	Client	Host	MW	SMSP
WS-REL-01	**Toolkit Implementations MUST ensure all relevant processing is completed before providing a SOAP response**	N	Y	Y	Y
1	• **Request Service Behaviour Type** - the request MUST be accepted and persisted. The “dummy” SOAP Response indicates the transfer of responsibility (in that the next recipient in the chain confirms that the message is received, accepted, and persisted) and confirms that this next recipient is now responsible for attempting further processing in due course.				

2	• **Request, Response Service Behaviour Type** – For an intermediary this means that the request MUST have been forwarded and all processing completed by the ultimate recipient application. The SOAP response indicates that all processing of the service request is complete, and contains any necessary information about the results.
3	• **Request, Response Service Behaviour Type** – For a recipient application this means that any update MUST have been committed to persistent storage. The SOAP response indicates that all processing of the service request is complete, and contains any necessary information about the results.
NB	In the case of the synchronous invocation, this SOAP response MUST be returned on an HTTP 200.
NB	In the case of the asynchronous invocation, this SOAP response will be returned via a separate HTTP connection over HTTP 202.

WS-REL-02	**Toolkit Implementations MAY retry if a transport (HTTP) response is not received**	**Y**	**N**	**Y**	**Y**
1	The lack of a transport-level (HTTP) response within the expected timeout period is an indication that the transport-level transmission may have failed. In these circumstances the sending Toolkit Implementation MAY retry.				

WS-REL-03	**Toolkit Implementations MUST retain responsibility for processing until a SOAP Response is received**	**Y**	**N**	**Y**	**Y**
NB	Specifically, the HTTP 202 response returned from the initial part of the asynchronous invocation pattern does NOT indicate a transfer of responsibility. It is only a transport acknowledgement, and it does NOT imply that the message has necessarily been persisted, nor promises that subsequent application processing will be completed.				

WS-REL-04	**The Toolkit Middleware MUST support configurable outbound retries**	**N**	**N**	**Y**	**N**

3.3 Validation

Ref	Description	Client	Host	MW	SMSP
WS-VAL-01	**The Toolkit Implementation SHOULD always perform at least syntactical validation before providing a SOAP response**	**N**	**Y**	**Y**	**Y**

3.4 Messaging Security

A combination of WS-Security SOAP headers and mutually-authenticated TLS are used to provide messaging security.

Ref	Description	Client	Host	MW	SMSP
WS-SEC-01	**Toolkit Implementations MUST use TLS for transport security. TLS 1.0 MUST be supported**	Y	Y	Y	Y
1	A valid certificate from a trusted CA MUST be used. It MUST provide for identification and authentication of the sender, by means of the “Subject” field in the certificate.				
2	TLS mutual authentication MUST be used.				
3	Connections MUST NOT be established if any of the following certificate errors occur when establishing a TLS session: • A Relying Party determines that the presented End Entity certificate ‘Subject’ ‘CN’ field does NOT match the FQDN of the End Entity presenting the certificate • A Relying Party cannot build a valid certificate path to validate the presented End Entity certificate to a trusted Root Certificate Authority that the Relying Party trusts • A Relying Party determines that any certificates in the certificate chain fail integrity checks • A Relying Party determines that any certificates in the certificate path are not yet valid, have expired or have been revoked.				
4	A certificate presented as part of TLS mutual authentication MAY have its subject Distinguished Name checked against an access control list or other reliable resource.				
5	In terms of version support, TLS 1.0 MUST be supported as a minimum, and higher levels (i.e. 1.1, 1.2) MAY also be supported. SSLv3 MAY also be supported, however SSLv2 MUST NOT be used.				

WS-SEC-02	**Toolkit Implementations MUST populate all messages with the identity of the requestor**	Y	N	Y	N
1	The Username element of the UsernameToken MUST be populated with the Subject CN value from the X.509 certificate used to secure the TLS connection.				

WS-SEC-04	**Toolkit Implementations MUST populate all messages with a Creation timestamp**	Y	N	Y	Y

WS-SEC-05	**Toolkit Implementations MUST populate all messages with an Expiry timestamp**	Y	N	Y	Y
NB	As a default 10 minutes is recommended.				

WS-SEC-06	**Toolkit Implementations MUST reject any message whose expiry timestamp has passed**	N	Y	Y	Y

WS-SEC-07	**Toolkit Implementations MUST be able to establish a requestor’s identity**	Y	Y	Y	Y

1	Toolkit Implementations MUST be able to use either TLS certificate subject or a WS-Security UsernameToken to establish a requestor's identity.
2	Although Toolkit implementations can choose a single method of requestor identification, they SHOULD be able to use both.
3	If implemented, a WS-Security UsernameToken SHOULD (except for the local security context) be verified against the Subject field CN of the presented X.509 certificate used to secure the TLS connection, if these 2 values are not equal, the Toolkit Implementation MUST reject the message.

WS-SEC-08	**Toolkit Implementations MUST be able to authorise a service request, based on the Service and the Requestor's identity**	**Y**	**Y**	**Y**	**Y**
NB	Inbound messages must be able to be checked to ensure that the requestor is indeed allowed to invoke this service. Due to the application-based security approach, the requestor's identity will usually be either that of the calling application or of an intermediary. As described in WS-SEC-07, for a Web Services implementation the identity of the requestor is provided by the Subject field of the certificate used to secure the TLS connection and / or the Username field of the WS-Security UsernameToken element.				
1	The Toolkit Implementation MUST include logic that ensures that the requestor is authorised to invoke that service e.g. via an Access Control List. The Toolkit Implementation MUST reject any messages that fail this authorisation check.				

WS-SEC-09	**Toolkit Implementations SHOULD have a flexible approach to certificate selection**	**Y**	**Y**	**Y**	**Y**
1	A sending system SHOULD be able to select between a number of available certificates when establishing a TLS connection.				
2	A receiving system SHOULD be able to configure multiple certificates in its trusted store.				

WS-SEC-16	**WS-Security elements MAY be omitted for SOAP messages returning on a synchronous HTTP Response**	**N**	**Y**	**Y**	**Y**
1	This requirement specifically overrides and takes precedence over all other requirements relating to WS-Security. In the case of a synchronous HTTP response down a channel already established by the sender then the WS-Security elements MAY be omitted.				

4 WS-Addressing

4.1 Message Identity - Element definitions

Element	Cardinality	Notes
MessageID	1..1	A unique identifier for each request message instance.

Table 4 : Element Definitions - Identity

4.2 Message Identity - Detailed Requirements

Ref	Description	Client	Host	MW	SMSP
WS-ADR-01	**The "MessageID" field MUST be populated with a unique uuid for each message instance**	**Y**	**Y**	**Y**	**Y**
1	Every web service call MUST have a unique MessageID formatted as a UUID.				
2	The UUID MUST be formatted into 5 hyphen-separated groups of hexadecimal digits having 8, 4, 4, 4, and 12 places respectively, and the hexadecimal digits A-F in UUIDs SHOULD be in upper case. Prefixes of "urn:" or "uuid:" MAY be used.				

WS-ADR-02	**There MUST NOT be any expectation that this MessageID is the same as other identifiers which may be defined within the message payload**	**Y**	**Y**	**Y**	**Y**

4.3 Addressing - Element definitions

Element	Cardinality	Notes
To	1..1	Indicates the url of the target endpoint.
Action	1..1	The SOAP Action being performed.
ReplyTo	0..1	Allows a url to be nominated for subsequent asynchronous SOAP response.
FaultTo	0..1	Allows a url to be nominated for subsequent asynchronous SOAP fault response.
RelatesTo	0..1	Used in a SOAP Response to correlate with the MessageID of the originating SOAP Request.

Table 5 : Element Definitions - Addressing

4.4 Addressing Detailed Requirements

Ref	Description	Client	Host	MW	SMSP
WS-ADR-03	**The “To” element MUST be populated with the url of the target endpoint**	**Y**	**Y**	**Y**	**Y**
1	For example, in the simple and common case of a SOAP message bound to a single-hop HTTP Post, the “To” element MUST contain the same HTTP address as the underlying HTTP Post.				
2	In the case of a synchronous request-response, then the response will be returned via the still-open HTTP connection. Thus the “To” address MAY NOT be supplied in this scenario. If, despite this, it is populated then it MUST contain the standard anonymous uri http://www.w3.org/2005/08/addressing/anonymous				

WS-ADR-04	**The “Action” element MUST be populated**	**Y**	**Y**	**Y**	**Y**
1	For a SOAP Request then the Action MUST be the same as the HTTP SOAP action, as specified in the WSDL HTTP binding.				
2	For a SOAP Response then the Action MUST be as per the corresponding SOAP Request, but with “Response” appended.				
3	For a SOAP Fault then the Action MUST be the standard action of http://www.w3.org/2005/08/addressing/soap/fault. For fundamental fault conditions such as (but not limited to) malformed XML, that are signalled synchronously, the Action MAY be omitted.				
4	For a WS-Addressing Fault then the Action MUST be the standard action of “http://www.w3.org/2005/08/addressing/fault.				
5	Synchronous faults MUST be returned with an HTTP 500 response code. Synchronous faults MAY exceptionally omit the WS-Addressing “Action” element where the fault is detected before a correct value for Action can be determined.				

WS-ADR-05	**The “From” element MAY be omitted**	**Y**	**Y**	**Y**	**Y**
1	Web Service Hosts MUST NOT require the “From” element to be populated.				
2	Web Service Clients MAY omit the FROM element.				

WS-ADR-06	**For asynchronous invocation the “ReplyTo” element of the request message MUST be populated with the uri to be used. (And the “FaultTo” element MAY be populated).**	**Y**	**Y**	**Y**	**Y**
1	In scenarios when a non-anonymous ReplyTo address is supplied then the caller MAY also supply a FaultTo address. This is an alternative address to be used if the asynchronous response is a fault. (For example, if the implementation comprises a separate fault handler).				
2	The ReplyTo address MUST contain a uri which is the address of the endpoint to be used for the SOAP response. (i.e. it MUST be the same as the “To” address which is subsequently used for the SOAP response).				

3	It is only relevant to supply this element in the initiating SOAP request message and MUST NOT be supplied in any other scenario (e.g. in the SOAP response messages themselves
4	In the case of a synchronous request-response, then the response will be returned via the still-open HTTP connection. Thus the ReplyTo address MAY NOT be supplied in this scenario. If, despite this, it is populated then it MUST contain the standard anonymous uri “http://www.w3.org/2005/08/addressing/anonymous”.
5	If the ReplyTo address is omitted then the recipient MUST treat this in exactly the same way as if an anonymous ReplyTo had been supplied.
6	The ReplyTo address MAY contain the uri of a queue, if “pull” collection of the response from a queue is desired.
NB	The ReplyTo address is needed to support the asynchronous invocation pattern, by indicating where the SOAP response will be sent.

WS-ADR-07	**For asynchronous invocation the “RelatesTo” element of the SOAP response MUST be populated with the MessageID of the initiating SOAP request message**	**Y**	**Y**	**Y**	**Y**
NB	The “RelatesTo” field in SOAP response message MUST contain the MessageID of the initiating SOAP request message, to assist the original requestor in correlating the asynchronous response.				

WS-ADR-08	**The ReferenceParameter and/or ReferenceProperty sub-elements of an Endpoint Reference MUST NOT be used**	**Y**	**Y**	**Y**	**Y**

WS-ADR-09	**For synchronous invocation, any response will be returned via the still-open HTTP connection. The RelatesTo field MAY be omitted in this scenario.**	**Y**	**Y**	**Y**	**Y**
NB	If, despite this, it is populated then it MUST contain the MessageID of the originating SOAP Request message.				

WS-ADR-10	**For synchronous invocation, any response will be returned via the still-open HTTP connection. Thus the ReplyTo address MAY NOT be supplied in this scenario.**	**Y**	**Y**	**Y**	**Y**
1	If, despite this, it is populated then it MUST contain the standard anonymous uri “http://www.w3.org/2005/08/addressing/anonymous”.				

2	If the ReplyTo address is omitted then the recipient MUST treat this in exactly the same way as if an anonymous ReplyTo had been supplied.

4.5 WS Security Time Stamp

Field	Cardinality	Notes
Created	1..1	A timestamp indicating when the message was created.
Expires	1..1	Provides the sender with an opportunity to specify a time beyond which the message should be considered as expired and not processed

Table 6 : Element Definitions – Time Stamp

Ref	Description	Client	Host	MW	SMSP
WS-DSC-01	**The "Created" timestamp element MUST be populated with the creation time of the message**				
1	Every web service call MUST have a fresh timestamp.				
NB	Deprecated. Found duplicate requirement of WS-SEC-04				

WS-DSC-02	**The "Expires" timestamp element MUST be populated with an expiry time for the message**				
NB	As a default 10 minutes is recommended.				
NB	Deprecated. Found duplicate requirement of WS-SEC-05				

WS-DSC-03	**All header timestamps MUST be in GMT/UTC and MUST be synchronised with a consistent time source to within 250 milliseconds**	**Y**	**Y**	**Y**	**Y**
1	All header timestamps MUST be synchronised with a time source common to all systems connected to that Toolkit instance. This time source SHOULD be the N3 NTP time service.				

4.6 WS-Security - Security Tokens

Field	Cardinality	Notes
Username	1..1	Used to identify application
Password	0..1	Not used

Table 7 : Element Definitions – Security Tokens

Ref	Description	Client	Host	MW	SMSP
WS-DSC-04	**A Username Token MUST be provided containing the identity of the application**	**Y**	**Y**	**Y**	**Y**

4.7 WS-Security - XML

Ref	Description	Client	Host	MW	SMSP
WS-DSC-17	**PKI certificates MUST be from a recognised CA**	**Y**	**Y**	**Y**	**Y**

WS-DSC-20	**Toolkit middleware and applications MAY preinstall the Root Certificate from other CAs that they choose to trust**	**Y**	**Y**	**Y**	**Y**

5 SOAP Header Extensibility

The design of SOAP headers is inherently intended to be flexible and extensible. To allow for potential future Toolkit enhancements it is important that this extensibility is realised by Toolkit implementations. Therefore:

Ref	Description	Client	Host	MW	SMSP
WS-EXT-01	**Implementations MUST understand all SOAP Header fields that are explicitly defined as mandatory in this specification document**	**Y**	**Y**	**Y**	**Y**

WS-EXT-02	**Implementations MUST be able to ignore any additional SOAP Header fields which are not in this specification, but which may be added in future**	**Y**	**Y**	**Y**	**Y**

WS-EXT-03	**Implementations MUST raise a SOAP Fault if unknown SOAP Header fields are encountered with the “mustUnderstand” attribute set**	**Y**	**Y**	**Y**	**Y**

WS-EXT-04	**Local SOAP Header extensions MUST use their own namespace**	**Y**	**Y**	**Y**	**Y**
1	SOAP Headers may be used to define local or implementation-specific extensions. In this case these MUST be clearly distinguished by the use of their own namespace.				
2	Local extensions SHOULD also define a schema for a single top-level container element – thus simplifying documentation and usage of these extensions.				

6 SOAP Faults

6.1 SOAP 1.1 Fault structure - Element definitions

Fault reporting is based on the standard SOAP 1.1 fault specification.

Element	Cardinality	Notes
faultcode	1..1	A code for identifying the fault. Must be one of the 4 standard SOAP fault codes.
faultstring	1..1	A human readable explanation of the fault. This MUST be populated, in accordance with the SOAP specification, but is anticipated as providing minimal information (and might be simple pre-defined text). The more useful application error information will be provided in the “detail” element.
faultactor	0..1	The endpoint where the fault occurred.
detail	0..1	Holds application specific error information related to the Body element. A Toolkit specific structure is defined for this detailed error information, see below for details.

Table 8 : Element Definitions – SOAP Fault Structure

6.1.1 Detailed Requirements

Ref	Description	Client	Host	MW	SMSP
WS-FLT-01	**The “faultcode” element MUST contain a standard fault code as-per SOAP and WS-* specifications**	**N**	**Y**	**Y**	**Y**

NB	SOAP defines 4 standard Fault Codes which are: • VersionMismatch - Found an invalid namespace for the SOAP Envelope element • MustUnderstand - An immediate child element of the Header element, with the mustUnderstand attribute set to "1", was not understood • Client - The message was incorrectly formed or contained incorrect information • Server - There was a problem with the server so the message could not proceed Refer to the SOAP specification for a more complete explanation of these standard codes Additional codes are defined by other WS-* specifications, for example WS-Addressing and WS-Security.

WS-FLT-02	**The “faultactor” element SHOULD be populated with the HTTP url of the endpoint raising the fault**	**N**	**Y**	**Y**	**Y**
NB	This may assist with tracing the source of the error.				

WS-FLT-03	**For faults reported due to “inability to complete” the request then the “detail” element MUST be populated with the Toolkit Fault structure**	**N**	**Y**	**Y**	**Y**
1	SOAP faults reported with a faultcode of “Client” or “Server” can arise either because of a malformed message or because of some other non-business failure-to-complete the request (for example, an authorisation failure or inability to contact a back-end service). Where the reason for the fault is such a failure-to-complete the SOAP fault “detail” element MUST be populated with a Toolkit Fault structure.				
2	This means that systems using third-party SOAP components which can fail a request early, for example for malformedness or a “MustUnderstand” error, and which generate their own SOAP faults under such circumstances, MAY replace the Toolkit Fault structure with their own appropriate content.				

6.2 SOAP 1.2 Fault Handling

The most significant changes made in SOAP 1.2 protocol specification is the fault handling block of the schema as opposed to its predecessor SOAP 1.1. Implementation of SOAP 1.2 has clear benefits over SOAP 1.1 . ITK specifications recommend using SOAP 1.2 wherever possible. Web service definition of the ITK Service can be implemented in both SOAP 1.1 and SOAP 1.2.

The key features of SOAP 1.2 fault schema are

- The elements are namespace qualified.
- The elements are hierarchically ordered.
- The Code and Reason elements are mandatory. The Node, Role and Detail elements are optional.

6.2.1 Detailed Requirements

To keep backward compatibility of ITK implementations, SOAP 1.2 fault handling requirement specifications are defined as follows:

Ref	Description	Client	Host	MW	SMSP
WS-FLT-21	**ITK Implementation MUST populate the <env:Code> element with at least one immediate child element <env:Value> . The value MUST be any one of the following and MUST be namespace qualified.** • VersionMismatch • MustUnderstand • DataEncodingUnknown • Sender • Receiver **ITK Implementation MAY implement <env:Subcode>**	Y	Y	Y	Y
NB	If the service endpoint binding used is SOAP 1.2 and the intention is not to change the application nested inside that expects a SOAP 1.1 fault structure, then the following mapping guidelines can be used. Please see more about the SOAP fault codes in the SOAP spec - fault codes section. ▪ <env:Value> can be mapped to SOAP 1.1 <env:Fault>/<faultcode> <env:Subcode> and subsequent child nodes e.g <env:Value> implementation is optional.				
WS-FLT-22	**ITK Implementation MUST populate the <env:Reason> element with a brief statement of the fault generated at the application level. The said statement MUST be wrapped within only one child element <env:Text xml:lang="">**	Y	Y	Y	Y
NB	The xml fragment of this block may look like this. <env:Reason> <env:Text xml:lang="en-GB ">Brief error description goes here</env:Text> </env:Reason> <env:Text> can be mapped to SOAP 1.1 <env:Fault>/<faultstring>				
WS-FLT-23	**ITK Implementation MUST populate the <env:Detail> element with <itk:ToolkitErrorInfo>**	Y	Y	Y	Y
NB	<env:Detail> can be mapped to SOAP 1.1 <env:Fault>/<detail> <env:Text> can be mapped to SOAP 1.1 <env:Fault>/<faultstring>				

WS-FLT-24	**ITK Implementation MAY implement <env:Node> and <env:Role>.**	**Y**	**Y**	**Y**	**Y**
NB	If implemented <env:Node> and <env:Role> can be mapped to SOAP 1.1 <env:Fault>/<faultactor>.				

6.3 Toolkit Fault structure - Element definitions

The following structure is defined for use in the SOAP Fault “detail” element, for transmitting error information between Toolkit applications.

Element	Cardinality	Notes
ErrorID	1..1	A unique id (uuid) to identify the error instance.
ErrorCode	1..1	An error code.
ErrorCode@codeSystem	0..1	An optional identifier (OID) indicating the vocabulary from which the error code comes. A simple generic set of default technical error codes is provided by HSCIC. It also is possible that alternative error vocabularies might be defined for specific services, should this be deemed necessary in future.
ErrorText	1..1	Additional error text provided by the application - suitable for displaying to a user (i.e. in addition to the standard text pre-defined with the error code).
ErrorDiagnosticText	0..1	Additional diagnostic information provided by the application, which might be of use to a systems administrator investigating the problem.

Table 9 : Element Definitions – Toolkit Fault Structure

6.4 Detailed Requirements

Ref	Description	Client	Host	MW	SMSP
WS-FDT-01	**The “ErrorID” element MUST be populated with a uuid to identify the error instance.**	**N**	**Y**	**Y**	**Y**
1	The uuid MUST be formatted into 5 hyphen-separated groups of hexadecimal digits having 8, 4, 4, 4, and 12 places respectively, and the hexadecimal digits A-F in UUIDs MUST be converted to upper case. The ErrorID MUST contain the uuid only. Specifically there MUST NOT be any prefixes such as “urn:” or “uuid:”.				

WS-FDT-02	**The “ErrorCode” element MUST be populated with an error code**	**N**	**Y**	**Y**	**Y**
NB	HSCIC have pre-defined the following error code vocabulary which SHOULD be used as the default (i.e. unless a different service-specific SOAP Fault error vocabulary is explicitly defined and documented). • **1000 = Invalid Message** (There is something incorrect with the message structure or content – a bug fix for the sender) • **2100 = Processing Error** (Retryable) (A processing error that may be retry-able – e.g. system temporarily down) • **2200 = Processing Erro**r (Not Retryable) (A processing error that is not retry-able – e.g. a programming bug in the integration infrastructure code) • **3000 = Access Denied** (A security problem)				

WS-FDT-03	**The “codeSystem” attribute MAY be populated with an OID that identifies an error code vocabulary**	**N**	**Y**	**Y**	**Y**
NB	The OID for the HSCIC default error code vocabulary described in HDR-FDT-02 is 2.16.840.1.113883.2.1.3.2.4.17.268 (ToolkitSOAPErrorCodes) - and this will be assumed if this attribute is omitted. If a different error vocabulary is used for some reason then the codeSystem attribute MUST be populated with an alternative OID to define what this is.				

WS-FDT-05	**The “ErrorText” element MUST be populated with additional human-readable description about the error – suitable for displaying to an end user**	**N**	**Y**	**Y**	**Y**

WS-FDT-06	**The “ErrorDiagnosticText” element SHOULD be populated with additional diagnostic information about the error**	**N**	**Y**	**Y**	**Y**

7 ITK Web Service in Practice

7.1 Synchronous Success Scenario

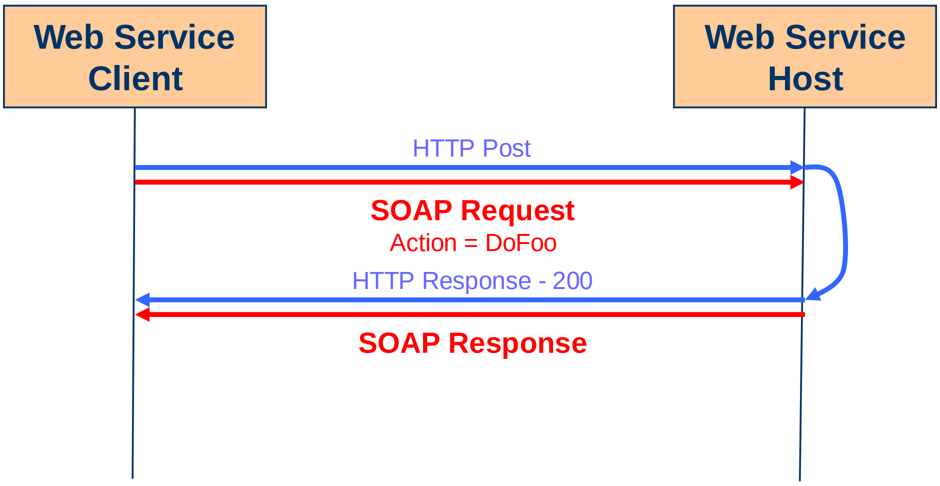


Figure 3 – Synchronous Success Scenario

a) The SOAP Request is bound to the HTTP Post

b) The Web Service Host completes processing, as appropriate for the relevant Toolkit Interaction / Messaging Configuration

c) The SOAP Response is bound to the HTTP 200 response

Note that "success" means that the web service call completes successfully. However it is possible that the SOAP Response contains information that indicates a business error.

7.2 Synchronous Failure Scenario 1

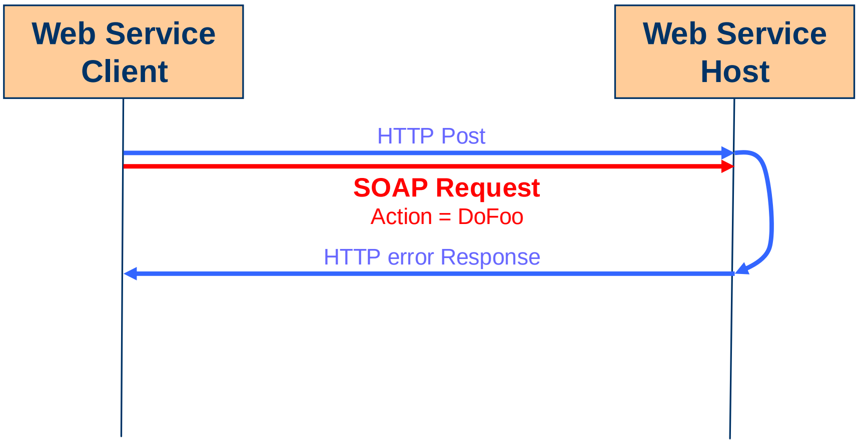


Figure 4 – Synchronous Failure Scenario I

a) The SOAP Request is bound to the HTTP Post

b) An error occurs at the HTTP level (e.g. server unavailable), causing an HTTP error response to be returned (e.g. 4xx, 5xx) with no SOAP payload

7.3 Synchronous Failure Scenario 2

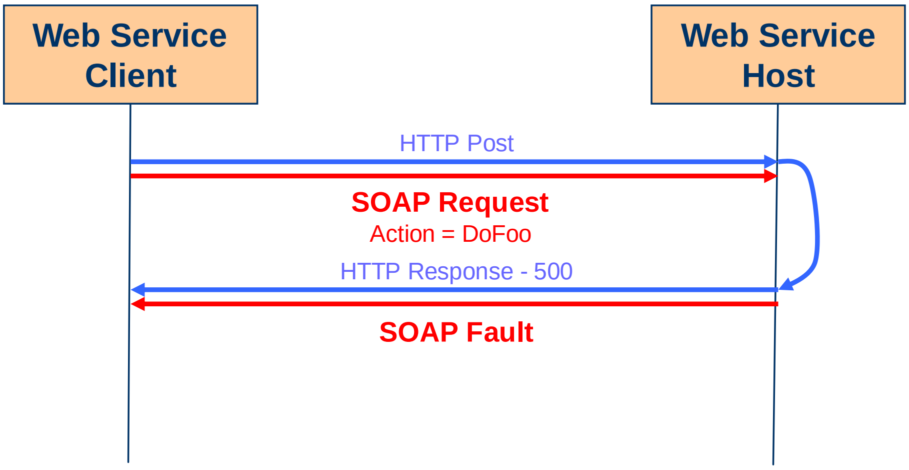


Figure 5 – Synchronous Failure Scenario II

a) The SOAP Request is bound to the HTTP Post

b) The Web Service Host encounters a technical error during processing.

c) The SOAP Fault is bound to the HTTP 500 response

7.4 Asynchronous Success Scenario

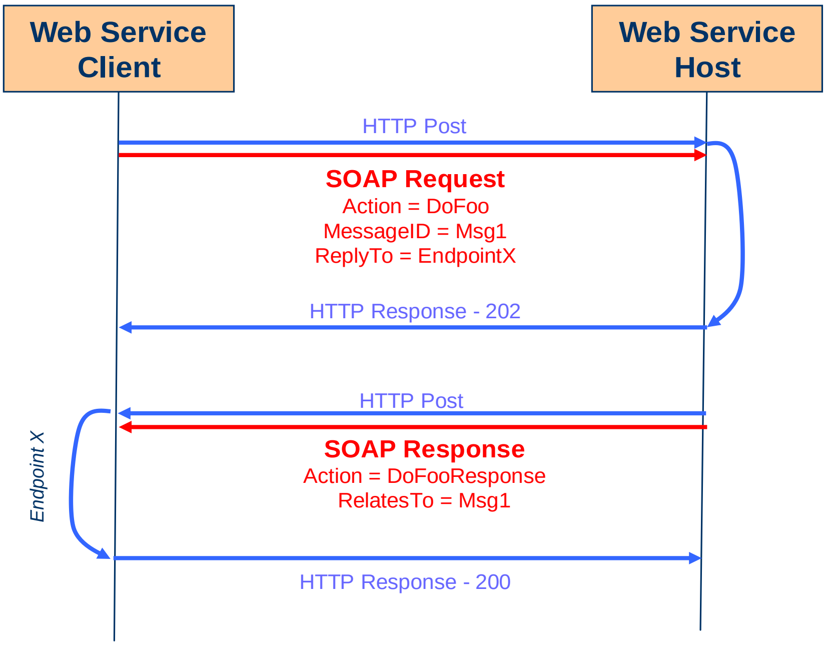


Figure 6 – Asynchronous Success Scenario

a) The SOAP Request is bound to the HTTP Post

b) The HTTP response is a 202 which SHOULD contain no SOAP payload. (It has been observed that some tools do put some default SOAP content in this response, in which case this MUST be ignored).

c) The Web Service Host completes processing, as appropriate for the relevant Toolkit Interaction / Messaging Configuration.

d) The Web Service Host opens a new HTTP connection to the endpoint specified in the “ReplyTo” header of the SOAP Request.

e) The SOAP Response is bound to the HTTP Post on this new connection.

f) The HTTP response is a 200 which again SHOULD contain no SOAP payload. (It has been observed that some tools do put some default SOAP content in this response, in which case this MUST be ignored).

Note that “success” means that the web service call completes successfully. However it is possible that the SOAP Response contains information that indicates a business error.

7.5 Asynchronous Failure Scenario 1

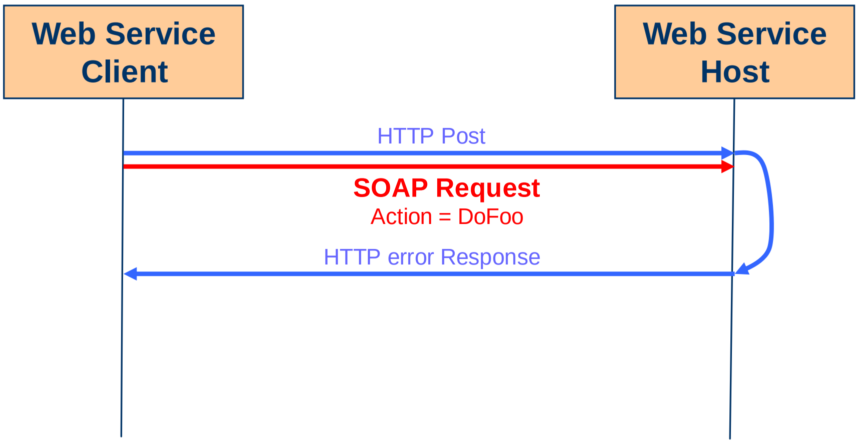


Figure 7 – Asynchronous failure Scenario I

a) The SOAP Request is bound to the HTTP Post

b) An error occurs at the HTTP level (e.g. server unavailable), causing an HTTP error response to be returned (e.g. 4xx, 5xx) with no SOAP payload

7.6 Asynchronous Failure Scenario 2

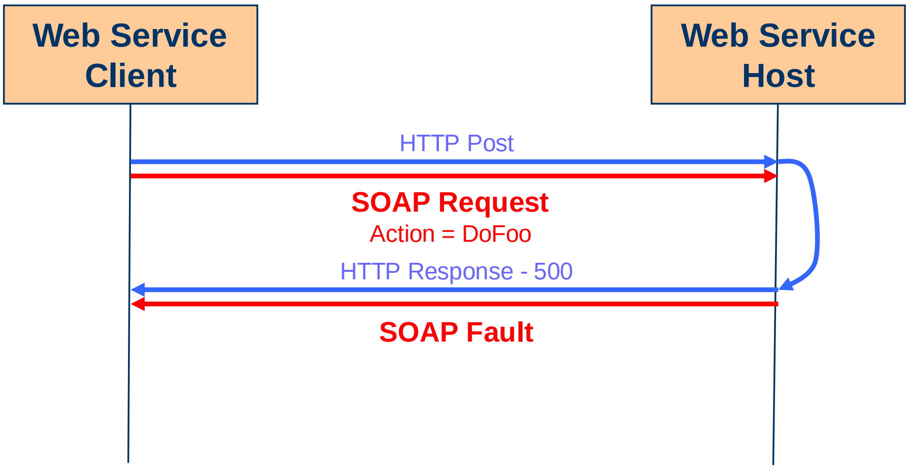


Figure 8 – Asynchronous failure Scenario II

a) The SOAP Request is bound to the HTTP Post

b) The Web Service Host encounters a technical error during processing prior to issuing the HTTP response. (For example unable to parse the incoming request).

c) A SOAP Fault is returned bound to a HTTP 500 response

7.7 Asynchronous Failure Scenario 3

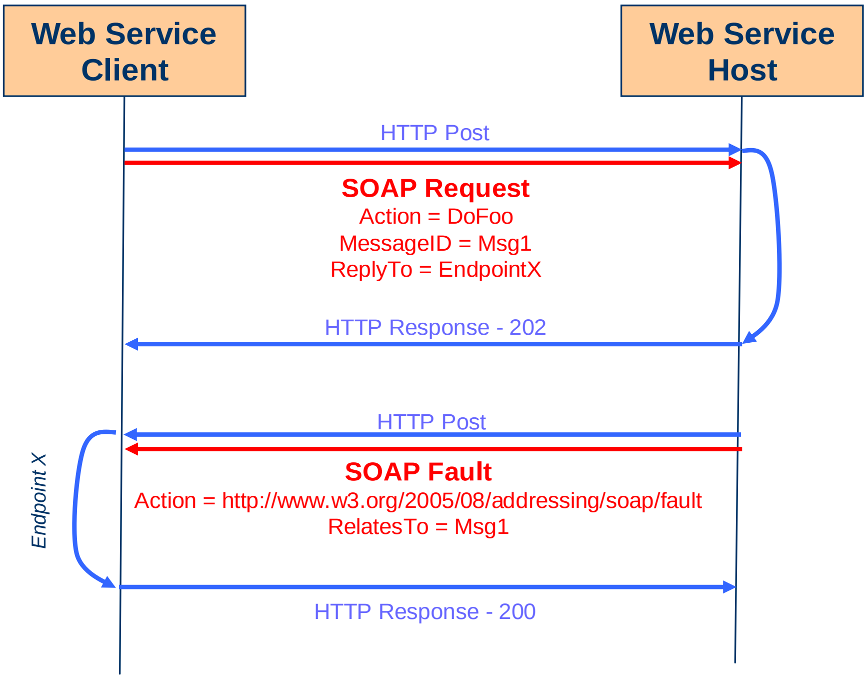


Figure 9 – Asynchronous failure Scenario III

a) The SOAP Request is bound to the HTTP Post

b) The HTTP response is a 202 which SHOULD contain no SOAP payload. (It has been observed that some tools do put some default SOAP content in this response, in which case this MUST be ignored).

c) The Web Service Host encounters a technical error during processing

d) The Web Service Host opens a new HTTP connection to the endpoint specified in the “ReplyTo” (or FaultTo) header of the SOAP Request. (See WS-ADR-06 for details)

e) The SOAP Fault is bound to the HTTP Post on this new connection.

f) The HTTP response is a 200 which again SHOULD contain no SOAP payload. (It has been observed that some tools do put some default SOAP content in this response, in which case this MUST be ignored).

Note that at Step f, there is also the possibility of a HTTP failure being returned. In this case the Web Service Host MAY retry and SHOULD log the error. However ultimately the responsibility lies with the Web Service Client which MUST timeout and initiate error handling if an asynchronous response is not received within an expected timeframe.

7.8 Asynchronous Failure Scenario 4

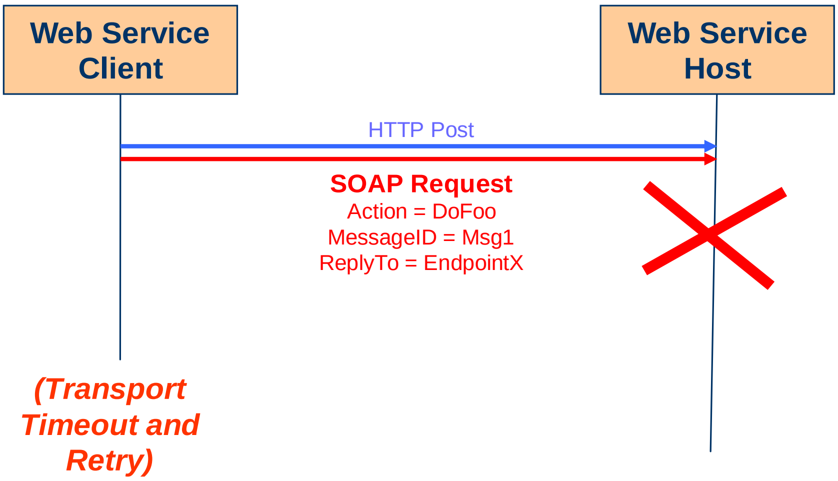


Figure 10 – Asynchronous failure Scenario IV

a) The SOAP Request is bound to the HTTP Post

b) An error occurs, causing the HTTP Post to timeout

c) The Web Service client will detect a transport level timeout, and MAY retry

(Note that this same failure scenario also applies to the synchronous case)

7.9 Asynchronous Failure Scenario 5

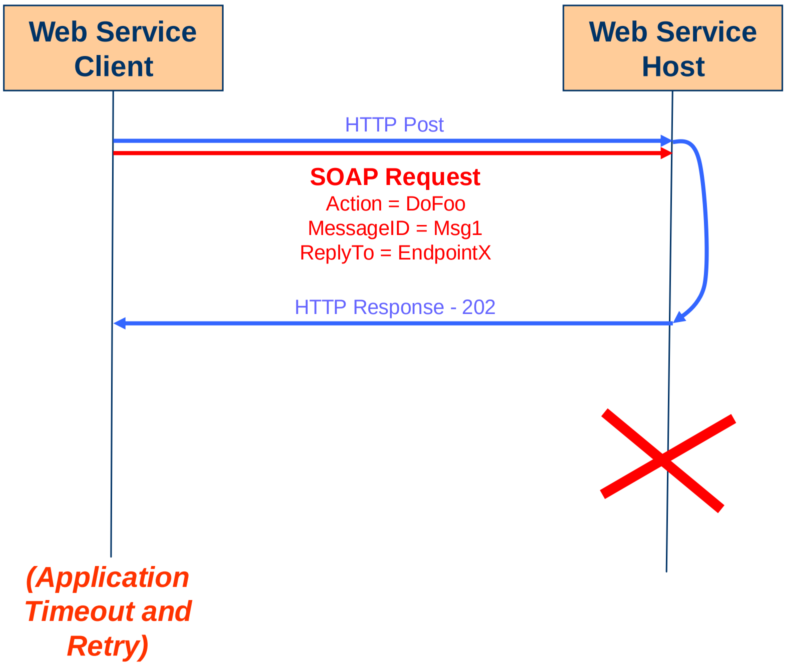


Figure 11 – Asynchronous failure Scenario V

a) The SOAP Request is bound to the HTTP Post

b) After issuing the HTTP 202 Response then an error occurs. As a result of this error it is not possible to return a SOAP Fault (e.g. server crash)

c) The Web Service Host has NOT necessarily persisted the message, and responsibility remains with the Web Service Client to detect an application-level timeout and perform appropriate error handling. For example it MAY retry.

7.10 SOAP Fault Handling Use Cases

A typical example of SOAP 1.2 fault out of ITK implementation is given below. Elements shown in **RED** are mandatory and that of **GREEN** are optional.

```
<env:Fault>
  <env:Code>
    <env:Value>env:Sender</env:Value>
    <env:Subcode>
      <env:Value>env:Sender</env:Value>
      <env:Subcode> <!-- Recursive Subcode is possible --> </env:Subcode>
    </env:Subcode>
  </env:Code>

  <env:Reason>
    <env:Text xml:lang="en-GB">Brief error description</env:Text>
    <!—- Multiple Text is possible but ITK specifies only one child node -->
  </env:Reason>

  <env:Node>http://nhs.net/theNodeThatFailed</env:Node>

  <env:Role>
    http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver
  </env:Role>

  <env:Detail>
    <itk:ToolkitErrorInfo xmlns:itk="urn:nhs-itk:ns:201005">
      <!—- application specific error -->
    </itk:ToolkitErrorInfo>
  </env:Detail>
</env:Fault>
```

Use Case I (SOAP 1.1 node receives SOAP 1.2)

If a SOAP 1.1 node receives a SOAP 1.2 request message is not processed. A typical SOAP fault may look as follows.

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header>
<!-- Header is optional -->
</env:Header>
<env:Body>

  <env:Fault xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
    <faultcode>env:VersionMismatch</faultcode>
    <faultstring>Version Mismatch: Message was not SOAP 1.1 conformant</faultstring>
    <faultactor>http://service.nhs.net/soap/actor/</faultactor>
  </env:Fault>

</env:Body>
```

Use Case II (SOAP 1.2 node receives SOAP 1.1)

If a SOAP 1.2 node receives a SOAP 1.1 request message may be processed or a typical SOAP fault may look as follows.

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xml="http://www.w3.org/XML/1998/namespace">
 <env:Header>
  <env:Upgrade>
   <env:SupportedEnvelope qname="ns1:Envelope"
             xmlns:ns1="http://www.w3.org/2003/05/soap-envelope"/>
   <env:SupportedEnvelope qname="ns2:Envelope"
             xmlns:ns2="http://schemas.xmlsoap.org/soap/envelope/"/>
  </env:Upgrade>
 </env:Header>
 <env:Body>
  <env:Fault>
   <env:Code><env:Value>env:VersionMismatch</env:Value></env:Code>
    <env:Reason>
      <env:Text xml:lang="en">Message was not SOAP 1.1 conformant </env:Text>
    </env:Reason>
   </env:Fault>
 </env:Body>
</env:Envelope>
```

* * * End of Document * * *