SOAP Version 1.2 Specification Assertions and Test Collection (Second Edition)

W3C Recommendation 27 April 2007

This version:
http://www.w3.org/TR/2007/REC-soap12-testcollection-20070427/
Latest version:
http://www.w3.org/TR/soap12-testcollection
Previous versions:
http://www.w3.org/TR/2006/PER-soap12-testcollection-20061219/
Editors:
Hugo Haas, W3C
Oisin Hurley, IONA Technologies
Anish Karmarkar, Oracle
Jeff Mischkinsky, Oracle
Mark Jones, AT&T
Lynne Thompson, Unisys
Richard Martin, Active Data Exchange

Please refer to the **errata** for this document, which may include normative corrections.

See also translations.

Abstract

This document draws on assertions found in the SOAP Version 1.2 specifications [SOAP Part 1], [SOAP Part 2], and provides a set of tests in order to show whether the assertions are implemented in a SOAP processor.

A SOAP 1.2 implementation that passes all of the tests specified in this document may claim to conform to the SOAP 1.2 Test Suite, 2007 04 27. It is incorrect to claim to be compliant with the SOAP Version 1.2 specifications merely by passing successfully all the tests provided in this test suite. It is also incorrect to claim that an implementation is non compliant with the SOAP Version 1.2 specifications based on its failure to pass one or more of the tests in this test suite.

Status of this Document

This section describes the status of this document at the time of its publication. Other *documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.*

This document is a W3C Recommendation. It has been produced by the XML Protocol Working Group, which is part of the Web Services Activity. This second edition updates and supersedes the original Recommendation by the inclusion of the accumulated errata. Changes between these two versions are described in a diff document.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

Please report errors in this document to the public mailing list xmlp-comments@w3.org (archive). It is inappropriate to send discussion email to this address.

SOAP Version 1.2 supercedes all previous versions of SOAP, including SOAP Version 1.1 [soap11]

The SOAP 1.2 Implementation Report can be found at http://www.w3.org/2000/xp/Group/2/03/soap1.2implementation.html. Additional implementation experience of the Request Optional Response MEP can be found in the WSDL 2.0 implementation testing here: http://dev.w3.org/cvsweb/~checkout~/2002/ws/desc/test-suite/Dashboard.html.

This document is governed by the 24 January 2002 CPP as amended by the W3C Patent Policy Transition Procedure. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

A list of current W3C Recommendations and other technical reports can be found at http://www.w3.org/TR.

Short Table of Contents

Table of Contents

1. Introduction

This document draws on assertions found in the SOAP Version 1.2 specifications, and provides a set of tests in order to show whether the assertions are implemented in a SOAP processor. The primary goal of this document is to foster interoperability between different SOAP 1.2 implementations. The document is intended to help implementors to write SOAP processors that comply with SOAP 1.2 specification, and interoperate with other SOAP processors that comply with SOAP 1.2 specification.

A SOAP 1.2 implementation that passes all of the tests specified in this document may claim to conform to the SOAP 1.2 Test Suite $Date 2007/04/27 $.

Even though the purpose of the SOAP 1.2 Test Suite is to facilitate the creation of interoperable implementations, conformance to the SOAP 1.2 Test Suite does not imply conformance to the SOAP 1.2 specifications; there are mandatory requirements of the specifications that are not tested by the suite (as a simple example, SOAP 1.2 requires that every legal value of a role name is accepted, and all illegal ones rejected). An implementation may be said to be SOAP 1.2 conformant if and only if it it satisfies the conformance requirements specified in SOAP 1.2 specifications. The W3C does not at this time provide for any comprehensive means of testing for such conformance.

Similarly, an implementation may conform to the SOAP 1.2 specifications even if it does not support all capabilities tested by the SOAP 1.2 Test Suite. SOAP 1.2 specifications admits special purpose implementations, such as those in dedicated controllers, which may send and receive only a very limited suite of messages; the requirement is that whatever is done be done correctly. An implementation may conform to the SOAP 1.2 specifications even if it does not support all capabilities tested by the SOAP 1.2 Test Suite. The test suite defines higher level application semantics to enable testing and facilitate interoperable implementations. It is not necessary for a SOAP processor to support these higher level semantics to be SOAP 1.2 compliant.

Assertions for SOAP Version 1.2 Part 1 and Part 2 are numbered sequentially (1..n). "Location of the assertion" points the source of the assertion (section or subsection number) in Part 1 or Part 2. Hyperlinks are used to cross-reference to the original specification section/subsection.

Some of the tests in this document use SOAPBuilders interoperability tests as a started point, but have been modified to conform to the SOAP 1.2 specifications.

2. SOAP 1.2 Assertions

2.1 SOAP 1.2, Part 1 Assertions

Assertion x1-conformance-part1

Location of the assertion

SOAP 1.2 Part 1, Section 1.2

Text from the specification

> For an implementation to claim conformance with the SOAP Version 1.2 specification, it MUST correctly implement all mandatory ("MUST") requirements expressed in Part 1 of the SOAP Version 1.2 specification (this document) that pertain to the activity being performed. Note that an implementation is not mandated to implement all the mandatory requirements.

Comments

This statement applies to all assertions and as such will not be tested separately.

Assertion x1-conformance-part2

Location of the assertion

SOAP 1.2 Part 1, Section 1.2

Text from the specification

> The implementation of an Adjunct MUST implement all the pertinent mandatory requirements expressed in the specification of the Adjunct to claim conformance with the Adjunct.

Comments

This statement applies to all assertions in part 2 and as such will not be tested separately.

Assertion x1-reltoxml-noschema

Location of the assertion

SOAP 1.2 Part 1, Section 1.3

Text from the specification

> SOAP does not require that XML Schema processing (assessment or validation) be performed to establish the correctness or 'schema implied' values of *element* and *attribute information items* defined by Parts 1 and 2 of this specification. The values associated with *element* and *attribute information items* defined in this specification MUST be carried explicitly in the transmitted SOAP message except where stated otherwise (see 5. SOAP Message Construct).

Comments

This assertion will not be tested.

Assertion x1-reltoxml-lexicalform

Location of the assertion

SOAP 1.2 Part 1, Section 1.3

Text from the specification

> Unless otherwise stated, all lexical forms are supported for each such attribute, and lexical forms representing the same value in the XML Schema value space are considered equivalent for purposes of SOAP processing, e.g. the boolean lexical forms "1" and "true" are interchangeable.

Comments

This assertion will not be tested.

Assertion x1-soapnodes-procmodel

Location of the assertion

SOAP 1.2 Part 1, Section 2.1

Text from the specification

> A SOAP node receiving a SOAP message MUST perform processing according to the SOAP processing model as described in this section and in the remainder of this specification.

Comments

This assertion is tested by the entire test collection.

Assertion x1-soaproles-invariant

Location of the assertion

SOAP 1.2 Part 1, Section 2.2

Text from the specification

> The roles assumed by a node MUST be invariant during the processing of an individual SOAP message.

Comments

This assertion cannot be fully tested, as a SOAP node is allowed to process and remove SOAP headers, reinsert them and send them upstream.

Tests

T62

Assertion x1-soaproles-next

Location of the assertion

SOAP 1.2 Part 1, Section 2.2

Text from the specification

> Table 2: SOAP Roles defined by this specification, row 1

Tests

T1, T17, T66, T67, T68, T74, T75, TH4

Assertion x1-soaproles-none

Location of the assertion

SOAP 1.2 Part 1, Section 2.2

Text from the specification

> Table 2: SOAP Roles defined by this specification, row 2

Tests

T8, T18, T19

Assertion x1-soaproles-ur

Location of the assertion

SOAP 1.2 Part 1, Section 2.2

Text from the specification

> Table 2: SOAP Roles defined by this specification, row 3

Tests

T36, T37, T78, T79

Assertion x1-muprocessing-true

Location of the assertion

SOAP 1.2 Part 1, Section 2.4

Text from the specification

> Mandatory SOAP header blocks are presumed to somehow modify the semantics of other SOAP header blocks or SOAP body elements. Therefore, for every mandatory SOAP header block targeted to a node, that node MUST either process the header block or not process the SOAP message at all, and instead generate a fault (see 2.6 Processing SOAP Messages and 5.4 SOAP Fault).

Comments

All tests in the test collection that use mustUnderstand attribute with a value of true/1 will test this assertion.

Assertion x1-muprocessing-ur

Location of the assertion

SOAP 1.2 Part 1, Section 2.4

Text from the specification

> In particular, it is not an error for an ultimate SOAP receiver to receive a message containing a mandatory SOAP header block that is targeted at a role other than the ones assumed by the ultimate SOAP receiver.

Tests

T15, T19

Assertion x1-structinterpbodies-ur

Location of the assertion

SOAP 1.2 Part 1, Section 2.5

Text from the specification

> An ultimate SOAP receiver MUST correctly process the immediate children of the SOAP body (see 5.3 SOAP Body).

Comments

All tests in the test collection that have body block(s).

Assertion x1-procsoapmsgs-steps

Location of the assertion

SOAP 1.2 Part 1, Section 2.6

Text from the specification

Unless otherwise stated, processing of all generated SOAP messages, SOAP faults and application-level side effects MUST be semantically equivalent to performing the following steps separately, and in the order given.

1. Determine the set of roles in which the node is to act. The contents of the SOAP envelope, including any SOAP header blocks and the SOAP body, MAY be inspected in making such determination.
2. Identify all header blocks targeted at the node that are mandatory.
3. If one or more of the SOAP header blocks identified in the preceding step are not understood by the node then generate a single SOAP fault with the `Value` of `Code` set to "env:MustUnderstand" (see 5.4.8 SOAP mustUnderstand Faults). If such a fault is generated, any further processing MUST NOT be done. Faults relating to the contents of the SOAP body MUST NOT be generated in this step.

   Note:

   Throughout this document, the term "`Value` of `Code` " is used as a shorthand for "value of the `Value` child *element information item* of the `Code` *element information item*" (see 5.4.1 SOAP Code Element).
4. Process all mandatory SOAP header blocks targeted at the node and, in the case of an ultimate SOAP receiver, the SOAP body. A SOAP node MAY also choose to process non-mandatory SOAP header blocks targeted at it.
5. In the case of a SOAP intermediary, and where the SOAP message exchange pattern and results of processing (e.g. no fault generated) require that the SOAP message be sent further along the SOAP message path, relay the message as described in section 2.7 Relaying SOAP Messages.

Comments

All tests in the test collection test this assertion.

Assertion x1-procsoapmsgs-headerspec

Location of the assertion

SOAP 1.2 Part 1, Section 2.6

Text from the specification

In all cases where a SOAP header block is processed, the SOAP node MUST understand the SOAP header block and MUST do such processing in a manner fully conformant with the specification for that header block.

Comments

All tests in the test collection that process a soap header without generating a fault, test this assertion.

Assertion x1-procsoapmsgs-body

Location of the assertion

SOAP 1.2 Part 1, Section 2.6

Text from the specification

> An ultimate SOAP receiver MUST process the SOAP body, in a manner consistent with 2.5 Structure and Interpretation of SOAP Bodies.

Comments

All tests in the test collection that have body block(s) test this assertion.

Assertion x1-procsoapmsgs-fault

Location of the assertion

SOAP 1.2 Part 1, Section 2.6

Text from the specification

> Failure is indicated by the generation of a fault (see 5.4 SOAP Fault). SOAP message processing MAY result in the generation of at most one fault.

Comments

All tests in the test collection that generate a fault test this assertion.

Assertion x1-procsoapmsgs-faultchoice

Location of the assertion

SOAP 1.2 Part 1, Section 2.6

Text from the specification

The selection of a fault need not be predicated on the application of the "MUST", "SHOULD" or "MAY" keywords to the generation of the fault, with the exception that if one or more of the prescribed faults is qualified with the "MUST" keyword, then any one fault from the set of possible faults MUST be generated.

Comments

This assertion will not be tested.

Assertion x1-relayable-behavior

Location of the assertion

SOAP 1.2 Part 1, Section 2.7.1

Text from the specification

Table 3: SOAP Nodes Forwarding behavior

Assertion x1-forwardinter-behavior

Location of the assertion

SOAP 1.2 Part 1, Section 2.7.2

Text from the specification

Forwarding SOAP intermediaries MUST process the message according to the SOAP processing model defined in 2.6 Processing SOAP Messages. In addition, when generating a SOAP message for the purpose of forwarding, they MUST:

1. Remove all processed SOAP header blocks.
2. Remove all non-relayable SOAP header blocks that were targeted at the forwarding node but ignored during processing.
3. Retain all relayable SOAP header blocks that were targeted at the forwarding node but ignored during processing.

Comments

All tests in the test collection that use Node B.

Assertion x1-forwardinter-featuresem

Location of the assertion

SOAP 1.2 Part 1, Section 2.7.2

Text from the specification

> Forwarding SOAP intermediaries MUST also obey the specification for the SOAP forwarding feature being used. The specification for such a feature MUST describe the required semantics, including the rules describing how the forwarded message is constructed.

Comments

This assertion will not be tested.

Assertion x1-soapinterminfoset-preserve

Location of the assertion

SOAP 1.2 Part 1, Section 2.7.4

Text from the specification

> All XML infoset properties of a message MUST be preserved with the following exceptions:
>
> 1. The *element information item* for a header block targeted at an intermediary MAY be removed, by that intermediary, from the [children] property of the SOAP `Header` *element information item* as detailed in 2.7.2 SOAP Forwarding Intermediaries.
> 2. *Element information items* for additional header blocks MAY be added to the [children] property of the SOAP `Header` *element information item* as detailed in 2.7.2 SOAP Forwarding Intermediaries.
> 3. *Whitespace character information items* MAY be removed from the [children] property of the SOAP `Envelope` *element information item*.
> 4. *Whitespace character information items* MAY be added to the [children] property of the SOAP `Envelope` *element information item*.
> 5. *Whitespace character information items* MAY be removed from the [children] property of the SOAP `Header` *element information item*.
> 6. *Whitespace character information items* MAY be added to the [children] property of the SOAP `Header` *element information item*.
> 7. *Comment information items* MAY be added to the [children] property of

the SOAP `Envelope` *element information item*.

8. *Comment information items* MAY be removed from the [children] property of the SOAP `Envelope` *element information item*.
9. *Comment information items* MAY be added to the [children] property of the SOAP `Header` *element information item*.
10. *Comment information items* MAY be removed from the [children] property of the SOAP `Header` *element information item*.
11. *Attribute information items* MAY be added to the [attributes] property of the SOAP `Envelope` *element information item*.
12. *Attribute information items* MAY be added to the [attributes] property of the SOAP `Header` *element information item*.
13. *Attribute information items* MAY be added to the [namespace attributes] property of the SOAP `Envelope` *element information item*.
14. *Attribute information items* MAY be added to the [namespace attributes] property of the SOAP `Header` *element information item*.
15. SOAP role *attribute information items* that are present in the [attributes] property of SOAP header block *element information items* may be transformed as described in 5.2.2 SOAP role Attribute.
16. SOAP mustUnderstand *attribute information items* that are present in the [attributes] property of SOAP header block *element information items* may be transformed as described in 5.2.3 SOAP mustUnderstand Attribute.
17. SOAP relay *attribute information items* that are present in the [attributes] property of SOAP header block *element information items* may be transformed as described in 5.2.4 SOAP relay Attribute.
18. The [base URI] property of the *document information item* need not be maintained.
19. The [base URI] property of *element information items* MAY be changed or removed.
20. The [character encoding scheme] property of the *document information item* MAY be changed or removed.
21. All *namespace information items* in the [in-scope namespaces] of *element information items* MUST be preserved. Additional namespace information items MAY be added.

Comments

This assertion is partially tested by all tests that use Node B.

Assertion x1-envvermodel-permsg

Location of the assertion

SOAP 1.2 Part 1, Section 2.8

Text from the specification

> A SOAP node determines whether it supports the version of a SOAP message on a per message basis.

Comments

This assertion will not be tested.

Assertion x1-envvermodel-msgversion

Location of the assertion

SOAP 1.2 Part 1, Section 2.8

Text from the specification

> A SOAP node MAY support multiple envelope versions. However, when processing a message, a SOAP node MUST use the semantics defined by the version of that message.

Tests

T34

Assertion x1-envvermodel-fault

Location of the assertion

SOAP 1.2 Part 1, Section 2.8

Text from the specification

> If a SOAP node receives a message whose version is not supported it MUST generate a fault (see 5.4 SOAP Fault) with a Value of Code set to "env:VersionMismatch". Any other malformation of the message construct MUST result in the generation of a fault with a Value of Code set to "env:Sender".

Tests

T24, T30, TH3, T14, T28, T69, T70, T71, T72, TH2

Assertion x1-soapfeature-override

Location of the assertion

SOAP 1.2 Part 1, Section 3.1

Text from the specification

> The processing of SOAP envelopes in accordance with the SOAP Processing Model (see 2. SOAP Processing Model) MUST NOT be overridden by binding specifications.

Comments

This assertion will not be tested.

Assertion x1-featurereq-spec

Location of the assertion

SOAP 1.2 Part 1, Section 3.1.1

Text from the specification

> The specification of a feature MUST include the following:
>
> 1. A URI used to name the feature. This enables the feature to be unambiguously referenced in description languages or during negotiation.
> 2. The information (state) required at each node to implement the feature.
> 3. The processing required at each node in order to fulfill the obligations of the feature including any handling of communication failures that might occur in the underlying protocol (see also 4.2 Binding Framework).
> 4. The information to be transmitted from node to node.

Comments

This assertion will not be tested.

Assertion x1-soapmep-spec

Location of the assertion

SOAP 1.2 Part 1, Section 3.2

Text from the specification

> The specification of a message exchange pattern MUST:
>
> - As mandated by 3.1.1 Requirements on Features, provide a URI to name the MEP.
> - Describe the life cycle of a message exchange conforming to the pattern.
> - Describe the temporal/causal relationships, if any, of multiple messages exchanged in conformance with the pattern (e.g. responses follow requests and are sent to the originator of the request.)
> - Describe the normal and abnormal termination of a message exchange conforming to the pattern.

Comments

This assertion will not be tested.

Assertion x1-soapmep-featurereq

Location of the assertion

SOAP 1.2 Part 1, Section 3.2

Text from the specification

> MEPs are SOAP features, so an MEP specification MUST conform to the requirements for SOAP feature specifications (see 3.1.1 Requirements on Features).

Comments

Assertion x1-soapmep-include

Location of the assertion

SOAP 1.2 Part 1, Section 3.2

Text from the specification

An MEP specification MUST also include:

1. Any requirements to generate additional messages (such as responses to requests in a request/response MEP).
2. Rules for the delivery or other disposition of SOAP faults generated during the operation of the MEP.

Comments

This assertion will not be tested.

Assertion x1-soapmodules-specrules

Location of the assertion

SOAP 1.2 Part 1, Section 3.3

Text from the specification

A module specification adheres to the following rules. It:

1. MUST identify itself with a URI. This enables the module to be unambiguously referenced in description languages or during negotiation.
2. MUST declare the features provided by a module (see 3.1 SOAP Features).
3. MUST clearly and completely specify the content and semantics of the SOAP header blocks used to implement the behavior in question, including if appropriate any modifications to the SOAP Processing model. The SOAP extensibility model does not limit the extent to which SOAP can be extended. Nor does it prevent extensions from modifying the SOAP processing model from that described in 2. SOAP Processing Model
4. MAY utilize the property conventions defined in [SOAP Part 2], section A Convention for Describing Features and Bindings, in describing the functionality that the module provides. If these conventions are followed, the module specification MUST clearly describe the relationship between the abstract properties and their representations in the SOAP envelope.

Note that it is possible to write a feature specification purely in terms of abstract properties, and then write a separate module specification which implements that feature, mapping the properties defined in the feature specification to SOAP header blocks in the SOAP module.

5. MUST clearly specify any known interactions with or changes to the interpretation of the SOAP body. Furthermore, it MUST clearly specify any known interactions with or changes to the interpretation of other SOAP features and SOAP modules For example, we can imagine a module which encrypts and removes the SOAP body, inserting instead a SOAP header block containing a checksum and an indication of the encryption mechanism used. The specification for such a module would indicate that the decryption algorithm on the receiving side is to be run *prior* to any other modules which rely on the contents of the SOAP body.

Comments

This assertion will not be tested.

Assertion x1-bindfw-enablemep

Location of the assertion

SOAP 1.2 Part 1, Section 4.2

Text from the specification

> A binding specification MUST enable one or more MEPs.

Comments

HTTP binding specified in SOAP 1.2 part 2 enables an MEP. This assertion will not be tested.

Assertion x1-bindfw-featurecomb

Location of the assertion

SOAP 1.2 Part 1, Section 4.2

Text from the specification

> In cases where multiple features are supported by a binding specification, the specifications for those features MUST provide any information necessary for their successful use in combination.

Comments

HTTP binding specified in SOAP 1.2 part 2 is an example of this assertion. This assertion will not be tested.

Assertion x1-bindfw-featuredepend

Location of the assertion

SOAP 1.2 Part 1, Section 4.2

Text from the specification

> Similarly, any dependencies of one feature on another (i.e. if successful use of one feature depends on use or non-use of another) MUST be specified.

Comments

HTTP binding specified in SOAP 1.2 part 2 is an example of this assertion. This assertion will not be tested.

Assertion x1-bindfw-xml10

Location of the assertion

SOAP 1.2 Part 1, Section 4.2

Text from the specification

> The binding framework does NOT require that every binding use the XML 1.0 [8] serialization as the "on the wire" representation of the Infoset; compressed, encrypted, fragmented representations and so on can be used if appropriate.

Comments

This assertion will not be tested.

Assertion x1-bindfw-streaming

Location of the assertion

SOAP 1.2 Part 1, Section 4.2

Text from the specification

> Although streaming SOAP receivers will acquire such Infosets incrementally, SOAP processing MUST yield results identical to those that would have been achieved if the entire SOAP envelope were available prior to the start of processing.

Comments

This assertion will not be tested.

Assertion x1-soapenv-diichild

Location of the assertion

SOAP 1.2 Part 1, Section 5

Text from the specification

> A SOAP message is specified as an XML infoset that consists of a *document information item* with exactly one member in its [children] property, which MUST be the SOAP `Envelope` *element information item* (see 5.1 SOAP Envelope). This *element information item* is also the value of the [document element] property.

Comments

All tests in the test collection test this assertion.

Assertion x1-soapenv-docprop

Location of the assertion

SOAP 1.2 Part 1, Section 5

Text from the specification

> The [notations] and [unparsed entities] properties are both empty. The [base URI], [character encoding scheme] and [version] properties may have any legal value. The [standalone] property either has a value of "true" or has no value.

Tests

T64, T25, T65, T66, T67

Assertion x1-soapenv-dtd

Location of the assertion

SOAP 1.2 Part 1, Section 5

Text from the specification

> The XML infoset of a SOAP message MUST NOT contain a *document type declaration information item*.

Tests

T25

Assertion x1-soapenv-pi

Location of the assertion

SOAP 1.2 Part 1, Section 5

Text from the specification

> SOAP messages sent by initial SOAP senders MUST NOT contain *processing instruction information items*. SOAP intermediaries MUST NOT insert *processing instruction information items* in SOAP messages they relay. SOAP receivers receiving a SOAP message containing a *processing instruction information item* SHOULD generate a SOAP fault with the `Value` of `Code` set to "env:Sender".

Tests

T26

Assertion x1-soapenvelope-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.1

Text from the specification

> The SOAP `Envelope` *element information item* has:
>
> - A [local name] of `Envelope` .
> - A [namespace name] of "http://www.w3.org/2003/05/soap-envelope".
> - Zero or more namespace qualified *attribute information items* amongst its [attributes] property.
> - One or two *element information item*s in its [children] property in order as follows:
>   1. An optional `Header` *element information item* (see 5.2 SOAP Header).
>   2. A mandatory `Body` *element information item* (see 5.3 SOAP Body).

Comments

All the tests in the test collection test this assertion.

Assertion x1-soapencattr-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.1.1

Text from the specification

> The `encodingStyle` *attribute information item* has:
>
> - A [local name] of `encodingStyle` .
> - A [namespace name] of "http://www.w3.org/2003/05/soap-envelope".
>
> The `encodingStyle` *attribute information item* is of type *xs:anyURI*.

Comments

All tests in the test collection that use encodingSytle attribute test this assertion.

Assertion x1-soapencattr-restrictions

Location of the assertion

SOAP 1.2 Part 1, Section 5.1.1

Text from the specification

> The `encodingStyle` *attribute information item* MAY appear on the following:
>
> 1. A SOAP header block (see 5.2.1 SOAP Header block).
> 2. A child *element information item* of the SOAP `Body` *element information item* (see 5.3.1 SOAP Body child Element) if that child is not a SOAP Fault *element information item* (see 5.4 SOAP Fault).
> 3. A child *element information item* of the SOAP `Detail` *element information item* (see 5.4.5.1 SOAP detail entry).
> 4. Any descendent of 1, 2, and 3 above.
>
> The `encodingStyle` *attribute information item* MUST NOT appear on any element other than above in a SOAP message infoset.

Comments

All tests in the test collection that use encodingSytle attribute test this assertion.

Assertion x1-soapencattr-scope

Location of the assertion

SOAP 1.2 Part 1, Section 5.1.1

Text from the specification

> The scope of the `encodingStyle` *attribute information item* is that of its owner *element information item* and that *element information item's* descendants, unless a descendant itself carries such an *attribute information item*.

Tests

T73

Assertion x1-soapencattr-none

Location of the assertion

SOAP 1.2 Part 1, Section 5.1.1

Text from the specification

> If no `encodingStyle` *attribute information item* is in scope for a particular *element information item* or the value of such an *attribute information item* is "http://www.w3.org/2003/05/soap-envelope/encoding/none" then no claims are made regarding the encoding style of that *element information item* and its descendants.

Tests

T73

Assertion x1-soaphead-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.2

Text from the specification

> The `Header` *element information item* has:
>
> - A [local name] of `Header` .
> - A [namespace name] of "http://www.w3.org/2003/05/soap-envelope".
> - Zero or more namespace qualified *attribute information items* in its [attributes] property.
> - Zero or more namespace qualified *element information item*s in its [children] property.

Comments

All the tests in the test collection that use headers, test this assertion.

Assertion x1-soapheadblock-prop

Location of the assertion

[SOAP 1.2 Part 1, Section 5.2.1]

Text from the specification

> Each SOAP header block *element information item*:
>
> - MUST have a [namespace name] property which has a value, that is the name of the element MUST be namespace qualified.
> - MAY have any number of *character information item* children. Child *character information items* whose character code is amongst the whitespace characters as defined by [XML 1.0] are considered significant.
> - MAY have any number of *element information item* children. Such *element information items* MAY be namespace qualified.
> - MAY have zero or more *attribute information items* in its [attributes] property. Among these MAY be any or all of the following, which have special significance for SOAP processing:
>   - `encodingStyle` *attribute information item* (see 5.1.1 SOAP encodingStyle Attribute).
>   - `role` *attribute information item* (see 5.2.2 SOAP role Attribute).
>   - `mustUnderstand` *attribute information item* (see 5.2.3 SOAP mustUnderstand Attribute).
>   - `relay` *attribute information item* (see 5.2.4 SOAP relay Attribute).

Comments

All tests in the test collection that use headers, test this assertion.

Assertion x1-soaprole-prop

Location of the assertion

[SOAP 1.2 Part 1, Section 5.2.2]

Text from the specification

> The `role` *attribute information item* has the following XML infoset properties:
>
> - A [local name] of `role` .
> - A [namespace name] of "http://www.w3.org/2003/05/soap-envelope".
> - A [specified] property with a value of "true".
>
> The type of the `role` *attribute information item* is *xs:anyURI*. The value of the `role` *attribute information item* is a URI that names a role that a SOAP node can assume.

Comments

All tests in the test collection that use roles, will test this assertion.

Assertion x1-soaprole-omit

Location of the assertion

SOAP 1.2 Part 1, Section 5.2.2

Text from the specification

> Omitting the SOAP `role` *attribute information item* is equivalent to supplying that attribute with a value of "http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver".

Tests

T3, T22, T32, T34, T35, T56, T57

Assertion x1-soaprole-acceptur

Location of the assertion

SOAP 1.2 Part 1, Section 5.2.2

Text from the specification

> SOAP senders SHOULD NOT generate, but SOAP receivers MUST accept the SOAP `role` *attribute information item* with a value of "http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver".

Tests

T36, T37

Assertion x1-soaprole-ignore

Location of the assertion

SOAP 1.2 Part 1, Section 5.2.2

Text from the specification

> A SOAP sender generating a SOAP message SHOULD use the `role` *attribute information item* only on SOAP header blocks. A SOAP receiver MUST ignore this *attribute information item* if it appears on descendants of a SOAP header block or on a SOAP body child *element information item* (or its descendents).

Tests

T74

Assertion x1-soapmu-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.2.3

Text from the specification

> The `mustUnderstand` *attribute information item* has the following XML infoset properties:
>
> - A [local name] of `mustUnderstand` .
> - A [namespace name] of "http://www.w3.org/2003/05/soap-envelope".
> - A [specified] property with a value of "true".
>
> The type of the `mustUnderstand` *attribute information item* is *xs:boolean*.

Comments

All tests in the test collection that use mustUnderstand attribute, test this assertion.

Assertion x1-soapmu-omit

Location of the assertion

SOAP 1.2 Part 1, Section 5.2.3

Text from the specification

> Omitting this *attribute information item* is defined as being semantically equivalent to including it with a value of "false".

Comments

'this' in the specification text refers to the mustUnderstand attribute information item.

Tests

T1, T2, T3, T4, T5, T6, T7, T9, T10, T18, T29, T37, T56, T57, T66, T67, T68, T74, T76

Assertion x1-soapmu-acceptfalse

Location of the assertion

SOAP 1.2 Part 1, Section 5.2.3

Text from the specification

> SOAP senders SHOULD NOT generate, but SOAP receivers MUST accept the SOAP `mustUnderstand` *attribute information item* with a value of "false" or "0".

Tests

T11, T38, T40,

Assertion x1-soapmu-allrep

Location of the assertion

SOAP 1.2 Part 1, Section 5.2.3

Text from the specification

> A SOAP receiver MUST accept any valid lexical representation of the attribute value.

Comments

The attribute mentioned in the spec text refers to env:mustUnderstand.

Tests

T11, T12, T13, T15, T16, T17, T19, T21, T22, T32, T35, T36, T38, T40, T62, T63, T74, T75, TH4

Assertion x1-soapmu-ignore

Location of the assertion

SOAP 1.2 Part 1, Section 5.2.3

Text from the specification

> A SOAP sender generating a SOAP message SHOULD use the `mustUnderstand` *attribute information item* only on SOAP header blocks. A SOAP receiver MUST ignore this *attribute information item* if it appears on descendants of a SOAP header block or on a SOAP body child *element information item* (or its descendents).

Comments

Assertion x1-soaprelay-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.2.4

Text from the specification

> The `relay` *attribute information item* has the following XML infoset properties:
>
> - A [local name] of `relay` .
> - A [namespace name] of "http://www.w3.org/2003/05/soap-envelope".
> - A [specified] property with a value of "true".
>
> The type of the relay *attribute information item* is *xs:boolean*.

Assertion x1-soaprelay-omit

Location of the assertion

SOAP 1.2 Part 1, Section 5.2.4

Text from the specification

> Omitting this *attribute information item* is defined as being semantically equivalent to including it with a value of "false".

Assertion x1-soaprelay-acceptfalse

Location of the assertion

SOAP 1.2 Part 1, Section 5.2.4

Text from the specification

> SOAP senders SHOULD NOT generate, but SOAP receivers MUST accept the SOAP `relay` *attribute information item* with a value of "false" or "0".

Assertion x1-soaprelay-allrep

Location of the assertion

SOAP 1.2 Part 1, Section 5.2.4

Text from the specification

> A SOAP receiver MUST accept any valid lexical representation of the attribute value.

Assertion x1-soaprelay-ignore

Location of the assertion

SOAP 1.2 Part 1, Section 5.2.4

Text from the specification

> A SOAP sender generating a SOAP message SHOULD use the `relay` *attribute information item* only on SOAP header blocks. A SOAP receiver MUST ignore this *attribute information item* if it appears on descendants of a SOAP header block or on a SOAP body child *element information item* (or its descendents).

Assertion x1-soapbody-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.3

Text from the specification

> The `Body` *element information item* has:
>
> - A [local name] of `Body` .
> - A [namespace name] of "http://www.w3.org/2003/05/soap-envelope".
> - Zero or more namespace qualified *attribute information items* in its [attributes] property.
> - Zero or more namespace qualified *element information item*s in its [children] property.

Comments

All tests in the test collection that have the Body element, test this assertion.

Assertion x1-soapfault-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4

Text from the specification

> The `Fault` *element information item* has:
>
> - A [local name] of `Fault` .
> - A [namespace name] of "http://www.w3.org/2003/05/soap-envelope".
> - Two or more child *element information item*s in its [children] property in order as follows:

1. A mandatory `Code` *element information item* (see 5.4.1 SOAP Code Element).
2. A mandatory `Reason` *element information item* (see 5.4.2 SOAP Reason Element).
3. An optional `Node` *element information item* (see 5.4.3 SOAP Node Element).
4. An optional `Role` *element information item* (see 5.4.4 SOAP Role Element).
5. An optional `Detail` *element information item* (see 5.4.5 SOAP Detail Element).).

Comments

All tests in the test collection that generate fault, test this assertion.

Assertion x1-soapfault-single

Location of the assertion

SOAP 1.2 Part 1, Section 5.4

Text from the specification

> To be recognized as carrying SOAP error information, a SOAP message MUST contain a single SOAP `Fault` *element information item* as the only child *element information item* of the SOAP `Body` .

Comments

All tests in the test collection that generate fault, test this assertion.

Assertion x1-soapfault-only

Location of the assertion

SOAP 1.2 Part 1, Section 5.4

Text from the specification

> When generating a fault, SOAP senders MUST NOT include additional *element information items* in the SOAP `Body` . A message whose `Body` contains a `Fault` plus additional *element information items* has no SOAP-defined semantics.

Comments

All tests in the test collection that generate fault, test this assertion.

Assertion x1-faultcodeelement-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.1

Text from the specification

> The `Code` *element information item* has:
>
> - A [local name] of `Code` .
> - A [namespace name] of `http://www.w3.org/2003/05/soap-envelope` .
> - One or two child *element information item*s in its [children] property, in order, as follows:
>   1. A mandatory `Value` *element information item* as described below (see 5.4.1.1 SOAP Value element (with Code parent))
>   2. An optional `Subcode` *element information item* as described below (see 5.4.1.2 SOAP Subcode element).

Comments

All tests in the test collection that generate fault, test this assertion.

Assertion x1-faultvalueelement-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.1.1

Text from the specification

> The `Value` *element information item* has:
>
> - A [local name] of `Value` .
> - A [namespace name] of `http://www.w3.org/2003/05/soap-envelope` .
>
> The type of the `Value` *element information item* is *env:faultCodeEnum*.

Comments

All tests in the test collection that generate fault, test this assertion.

Assertion x1-faultsubcodeelement-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.1.2

Text from the specification

The `Subcode` *element information item* has:

- A [local name] of `Subcode` .
- A [namespace name] of `http://www.w3.org/2003/05/soap-envelope` .
- One or two child *element information items* in its [children] property, in order, as follows:
  1. A mandatory `Value` *element information item* as described below (see 5.4.1.3 SOAP Value element (with Subcode parent)).
  2. An optional `Subcode` *element information item* (see 5.4.1.2 SOAP Subcode element).

Tests

T33, T80

Assertion x1-faultsubvalueelem-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.1.3

Text from the specification

The `Value` *element information item* has:

- A [local name] of `Value` .
- A [namespace name] of `http://www.w3.org/2003/05/soap-envelope` .

The type of the `Value` *element information item* is *xs:QName*.

Tests

T33, T80

Assertion x1-faultstringelement-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.2

Text from the specification

> The `Reason` *element information item* has:
>
> - A [local name] of `Reason` .
> - A [namespace name] of `http://www.w3.org/2003/05/soap-envelope` .
> - One or more `Text` *element information item* children (see 5.4.2.1 SOAP Text Element). Each child `Text` *element information item* SHOULD have a different value for its `xml:lang` *attribute information item*.
>
> The type of the `Reason` *element information item* is *env:faultReason*.

Comments

All tests in the test collection that generate fault, test this assertion.

Assertion x1-reasontextelement-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.2.1

Text from the specification

> The `Text` *element information item* has:
>
> - A [local name] of `Text` .
> - A [namespace name] of `http://www.w3.org/2003/05/soap-envelope` .
> - A mandatory *attribute information item* with a [local name] of `lang` and [namespace name] of "http://www.w3.org/XML/1998/namespace". Note that the definition in of the `lang` *attribute information item* requires that the [prefix] is "xml" or any capitalization thereof (see [XML 1.0], Language Identification).
> - Any number of *character information item* children. Child *character information items* whose character code is amongst the whitespace characters as defined by [XML 1.0] are considered significant.
>
> The type of the `Text` *element information item* is *env:reasontext*

Assertion x1-faultactorelement-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.3

Text from the specification

> The `Node` *element information item* has:
>
> - A [local name] of `Node` .
> - A [namespace name] of `http://www.w3.org/2003/05/soap-envelope` .
>
> The type of the `Node` *element information item* is *xs:anyURI*.

Tests

T21

Assertion x1-faultactorelement-intermediaries

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.3

Text from the specification

> SOAP nodes that do not act as the ultimate SOAP receiver MUST include this element information item.

Comments

The element information item in the specification text refers to the 'Node' element.

Tests

T21

Assertion x1-faultroleelement-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.4

Text from the specification

> The `Role` *element information item* has:
>
> - A [local name] of `Role` .
> - A [namespace name] of `http://www.w3.org/2003/05/soap-envelope` .
>
> The type of the `Role` *element information item* is *xs:anyURI*.
> The value of the `Role` *element information item* MUST be one of the roles assumed by the node during processing of the message (see 2.2 SOAP Roles and SOAP Nodes).

Tests

T21, TH4

Assertion x1-faultdetailelement-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.5

Text from the specification

> The `Detail` *element information item* has:
>
> - A [local name] of `Detail` .
> - A [namespace name] of `http://www.w3.org/2003/05/soap-envelope` .
> - Zero or more *attribute information item*s in its [attributes] property.
> - Zero or more child *element information item*s in its [children] property.

Tests

T27, T28, T58

Assertion x1-faultcodes-mu

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.6

Text from the specification

> A SOAP node MUST understand all SOAP fault codes in a SOAP fault message in order to be able to interpret the `Detail` *element information item* in a SOAP fault.

Comments

This assertion will not be tested

Assertion x1-soapupgrade-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.7.1

Text from the specification

> The `Upgrade` *element information item* has:
>
> - A [local name] of `Upgrade` .
> - A [namespace name] of "http://www.w3.org/2003/05/soap-envelope".
> - One or more `SupportedEnvelope` *element information item*s in its [children] property in 5.4.7.2 SOAP SupportedEnvelope Element.
>
> The `Upgrade` *element information item* MUST NOT have an `encodingStyle` *attribute information item*.

Tests

T30, TH3

Assertion x1-soapsupportedenv-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.7.2

Text from the specification

The `SupportedEnvelope` *element information item* has:

- A [local name] of `SupportedEnvelope` .
- A [namespace name] of "http://www.w3.org/2003/05/soap-envelope".
- A `qname` *attribute information item* in its [attributes] property as described in 5.4.7.3 SOAP QName Attribute.

Tests

T30, TH3

Assertion x1-soapqnamesu-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.7.3

Text from the specification

The `qname` *attribute information item* has the following XML infoset properties:

- A [local name] of `qname` .
- A [namespace name] which has no value.
- A [specified] property with a value of "true".

The type of the `qname` *attribute information item* is *xs:QName*.

Tests

T30, TH3

Assertion x1-soapnotunderstood-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.8.1

Text from the specification

Each `NotUnderstood` header block *element information item* has:

- A [local name] of `NotUnderstood` .
- A [namespace name] of "http://www.w3.org/2003/05/soap-envelope".
- A `qname` *attribute information item* in its [attributes] property as described in 5.4.8.2 SOAP QName Attribute.

The `NotUnderstood` *element information item* MUST NOT have an `encodingStyle` *attribute information item*.

Tests

T12, T13, T16, T17, T21, T35, T36, TH4,

Assertion x1-soapqnamenu-prop

Location of the assertion

SOAP 1.2 Part 1, Section 5.4.8.2

Text from the specification

The `qname` *attribute information item* has the following XML infoset properties:

- A [local name] of `qname` .
- A [namespace name] which has no value.
- A [specified] property with a value of "true".

The type of the `qname` *attribute information item* is *xs:QName*.

Tests

T12, T13, T16, T17, T21, T35, T36, TH4,

Assertion x1-useofuris-baseuri

Location of the assertion

SOAP 1.2 Part 1, Section 6

Text from the specification

SOAP does not define a base URI but relies on the mechanisms defined in [XML Base] and [RFC 3986] for establishing a base URI against which relative URIs can be made absolute.

Comments

seems to me that this assertion should be removed: ASK

Tests

T75

Assertion x1-useofuris-ipv6

Location of the assertion

SOAP 1.2 Part 1, Section 6

Text from the specification

> The use of IP addresses in URIs SHOULD be avoided whenever possible (see [RFC 1900]. However, when used, the literal format for IPv6 addresses in URIs as described by [RFC 3986] SHOULD be supported.

Comments

seems to me that this assertion should be removed: ASK

Tests

T40

Assertion x1-useofuris-length

Location of the assertion

SOAP 1.2 Part 1, Section 6

Text from the specification

> Any SOAP node MUST be able to handle the length of any URI that it publishes and both SOAP senders and SOAP receivers SHOULD be able to deal with URIs of at least 2048 characters in length.

Tests

T29

Assertion x1-version-soap11

Location of the assertion

SOAP 1.2 Part 1, Appendix A

Text from the specification

> If a SOAP node supports versioning from SOAP 1.1 to SOAP 1.2, then the SOAP node MUST implement the rules described in this appendix.
>
> The rules for dealing with the possible SOAP/1.1 and SOAP Version 1.2 interactions are as follows:
>
> 1. A SOAP/1.1 node receiving a SOAP Version 1.2 message will according to SOAP/1.1 generate a version mismatch SOAP fault based on a SOAP/1.1 message construct. That is, the envelope will have a [local name] of `Envelope` and a [namespace name] of "http://schemas.xmlsoap.org/soap/envelope/".
> 2. A SOAP Version 1.2 node receiving a SOAP/1.1 message either:
>   - MAY process the message as a SOAP/1.1 message (if supported), or
>   - MUST generate a version mismatch SOAP fault based on a SOAP/1.1 message construct following SOAP/1.1 semantics using a SOAP/1.1 binding to the underlying protocol. The SOAP fault SHOULD include an `Upgrade` SOAP header block as defined in this specification (see 5.4.7 VersionMismatch Faults) indicating support for SOAP Version 1.2. This allows a receiving SOAP/1.1 node to correctly interpret the SOAP fault generated by the SOAP Version 1.2 node.

Comments

The requirement on the behavior of SOAP 1.1 compliant SOAP node will not be tested by the test collection.

Tests

T30

2.2 SOAP 1.2, Part 2 Assertions

Assertion x2-values-struct

Location of the assertion

SOAP 1.2 Part 2, Section 2.3

Text from the specification

> A graph node whose outbound edges are distinguished solely by their labels is known as a "struct". The outbound edges of a struct MUST be labeled with distinct names (see 2.1.1 Edge labels).

Assertion x2-values-array

Location of the assertion

SOAP 1.2 Part 2, Section 2.3

Text from the specification

> A graph node whose outbound edges are distinguished solely by position is known as an "array". The outbound edges of an array MUST NOT be labeled.

Assertion x2-encrules-rep

Location of the assertion

SOAP 1.2 Part 2, Section 3.1

Text from the specification

> When serializing a graph for transmission inside a SOAP message, a representation that deserializes to the identical graph MUST be used; when multiple such representations are possible, any of them MAY be used. When receiving an encoded SOAP message, all representations MUST be accepted.

Tests

T76

Assertion x2-encodingedgesandnodes-edge

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.1

Text from the specification

> Each graph edge is encoded as an element information item and each element information item represents a graph edge.

Comments

All tests in the test collection that use soap encoding, will test this assertion.

Assertion x2-encodingedgesandnodes-edgeterm

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.1

Text from the specification

> The graph node at which an edge terminates is determined by examination of the serialized XML as follows:
>
> 1. If the *element information item* representing the edge does not have a `ref` *attribute information item* (see 3.1.5.2 ref Attribute Information Item) among its attributes then that *element information item* is said to *represent* a node in the graph and the edge terminates at that node. In such cases the *element information item* represents both a graph edge and a graph node
> 2. If the *element information item* representing the edge does have a `ref` *attribute information item* (see 3.1.5.2 ref Attribute Information Item) among its attributes, then the value of that *attribute information item* MUST be identical to the value of exactly one `id` *attribute information item* (see 3.1.5.1 id Attribute Information Item) in the same envelope. In this case the edge terminates at the graph node represented by the *element information item* on which the `id` *attribute information item* appears. That *element information item* MUST be in the scope of an `encodingStyle` attribute with a value of "http://www.w3.org/2003/05/soap-encoding" (see SOAP 1.2 Part 1 SOAP encodingStyle Attribute).
>
> All nodes in the graph are encoded as described in 1 above. Additional inbound edges for multi reference graph nodes are encoded as described in 2

above.

Comments

All tests in the test collection that use soap encoding and do not return a fault, test this assertion.

Assertion x2-simpleenc-lexvalue

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.2

Text from the specification

> The lexical value of a graph node representing a simple value is the sequence of Unicode characters identified by the *character information item* children of the *element information item* representing that node.

Tests

T41, T42, T43, T44, T45, T46, T47, T48, T49, T50, T51, T52, T53, T54, T55

Assertion x2-complexenc-obedge

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.3

Text from the specification

> An outbound edge of a graph node is encoded as an element information item child of the element information item that represents the node (see 3.1.1 Encoding graph edges and nodes).

Tests

T41, T42, T43, T44, T45, T46

Assertion x2-complexenc-distlab

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.3

Text from the specification

> For a graph edge which is distinguished by label, the [local name] and [namespace name] properties of the child *element information item* together determine the value of the edge label.

Tests

T41, T42, T43, T44, T45, T46

Assertion x2-complexenc-distpos

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.3

Text from the specification

> For a graph edge which is distinguished by position:
>
> - The ordinal position of the graph edge corresponds to the position of the child *element information item* relative to its siblings
> - The [local name] and [namespace name] properties of the child *element information item* are not significant.

Tests

T46, T47, T48, T49, T50

Assertion x2-complexenc-array

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.3

Text from the specification

> The following rules apply to the encoding of a graph node that represents an "array":
>
> - The element information item representing an array node MAY have amongst its attributes an itemType attribute information item (see 3.1.4.1 itemType Attribute Information Item).

> - The element information item representing an array node MAY have amongst its attributes an arraySize attribute information item (see 3.1.6 arraySize Attribute Information Item).

Comments

All tests in the test collection that use arrays, will test this assertion

Assertion x2-complexenc-nil

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.3

Text from the specification

> If a graph edge does not terminate in a graph node then it can either be omitted from the serialization or it can be encoded as an element information item with an xsi:nil attribute information item.

Tests

T77

Assertion x2-enctypename-tnprop

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.4

Text from the specification

> The type name property of a graph node is a {namespace name, local name} pair computed as follows:
>
> 1. If the *element information item* representing the graph node has an `xsi:type` *attribute information item* among its attributes then the type name property of the graph node is the value of the `xsi:type` *attribute information item*.
>
>   **Note:**
>
>   This attribute is of type *xs:QName* (see XML Schema [XML Schema Part 2]); its value consists of the pair {namespace name, local name}. Neither the prefix used to construct the QName nor any information relating to any definition of the type is considered to be part of the value. The SOAP graph carries only the qualified name of

the type.

2. Otherwise if the parent *element information item* of the *element information item* representing the graph node has an `enc:itemType` *attribute information item* (see 3.1.4.1 itemType Attribute Information Item) among its attributes then the type name property of the graph node is the value of the `enc:itemType` *attribute information item*
3. Otherwise the value of the type name property of the graph node is unspecified.

Comments

All encoding tests in the test collection, test this assertion.

Assertion x2-itemtypeattr-prop

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.4.1

Text from the specification

> The `itemType` *attribute information item* has the following Infoset properties:
>
> - A [local name] of `itemType` .
> - A [namespace name] of "http://www.w3.org/2003/05/soap-encoding".
> - A [specified] property with a value of "true".
>
> The type of the `itemType` *attribute information item* is *xs:QName*.

Tests

T27, T42, T46, T47, T48, T50, T58, T59, T60, T61

Assertion x2-idattr-prop

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.5.1

Text from the specification

> The `id` *attribute information item* has the following Infoset properties:
>
> - A [local name] of `id` .
> - A [namespace name] of "http://www.w3.org/2003/05/soap-encoding".

- A [specified] property with a value of "true".

The type of the `id` *attribute information item* is *xs:ID*.

Tests

T56, T57, T76

Assertion x2-refattr-prop

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.5.2

Text from the specification

The `ref` *attribute information item* has the following Infoset properties:

- A [local name] of `ref` .
- A [namespace name] of "http://www.w3.org/2003/05/soap-encoding".
- A [specified] property with a value of "true".

The type of the `ref` *attribute information item* is *xs:IDREF*.

Tests

T56, T57, T76

Assertion x2-uniqueidconstraints-onlyone

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.5.3

Text from the specification

The value of a `ref` *attribute information item* MUST also be the value of exactly one `id` *attribute information item*.

Tests

T56, T57, T76

Assertion x2-uniqueidconstraints-exclusive

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.5.3

Text from the specification

> A `ref` *attribute information item* and an `id` *attribute information item* MUST NOT appear on the same *element information item*.

Tests

T59

Assertion x2-arraySizeattr-prop

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.6

Text from the specification

> The `arraySize` *attribute information item* has the following Infoset properties:
>
> - A [local name] of `arraySize` .
> - A [namespace name] of "http://www.w3.org/2003/05/soap-encoding".
>
> The type of the `arraySize` *attribute information item* is *enc:arraySize*. The value of the `arraySize` *attribute information item* MUST conform to the following EBNF grammar

```
[1]  arraySizeValue    ::=  ("*" | concreteSize) nextConcreteSize*
[2]  nextConcreteSize  ::=  white space concreteSize
[3]  concreteSize      ::=  [0-9]+
[4]  white space       ::=  (#x20 | #x9 | #xD | #xA)+
```

> The array's dimensions are represented by each item in the list of sizes (unspecified size in case of the asterisk). The number of items in the list represents the number of dimensions in the array. The asterisk, if present, MUST only appear in the first position in the list. The default value of the `arraySize` *attribute information item* is "*", that is by default arrays are considered to have a single dimension of unspecified size.

Tests

T42, T46, T47, T48, T49, T50, T58, T60, T61

Assertion x2-nodeTypeattr-prop

Location of the assertion

SOAP 1.2 Part 2, Section 3.1.7

Text from the specification

> The `nodeType` *attribute information item* has the following Infoset properties:
>
> - A [local name] of `nodeType` .
> - A [namespace name] of "http://www.w3.org/2003/05/soap-encoding".
> - A [specified] property with a value of "true".
>
> The type of the `nodeType` *attribute information item* is *enc:nodeType*.
>
> The value of the `nodeType` *attribute information item* MUST, if present, be one of the strings "simple" or "struct" or "array". The value indicates what kind of a value this node represents - a simple value, a compound struct value or a compound array value respectively.

Assertion x2-soapforrpc-sdm

Location of the assertion

SOAP 1.2 Part 2, Section 4

Text from the specification

> The SOAP `encodingStyle` attribute information item (see SOAP 1.2 Part 1 [SOAP Part 1] SOAP encodingStyle Attribute) is used to indicate the encoding style of the RPC representation. The encoding thus specified MUST support the 2. SOAP Data Model.

Comments

This assertion will not be tested.

Assertion x2-rpcinvocation-model

Location of the assertion

SOAP 1.2 Part 2, Section 4.2.1

Text from the specification

An RPC invocation is modeled as a follows:

- The invocation is represented by a single struct containing an outbound edge for each [in] or [in/out] parameter. The struct is named identically to the procedure or method name and the conventions of B. Mapping Application Defined Names to XML Names SHOULD be used to represent method names that are not legal XML names.
- Each outbound edge has a label corresponding to the name of the parameter. The conventions of B. Mapping Application Defined Names to XML Names SHOULD be used to represent parameter names that are not legal XML names.

Comments

All tests in the test collection that use the RPC convention, test this assertion.

Assertion x2-rpcresponse-model

Location of the assertion

SOAP 1.2 Part 2, Section 4.2.2

Text from the specification

An RPC response is modeled as a follows:

- The response is represented by a single struct containing an outbound edge for the return value and each [out] or [in/out] parameter. The name of the struct is not significant.
- Each parameter is represented by an outbound edge with a label corresponding to the name of the parameter. The conventions of B. Mapping Application Defined Names to XML Names SHOULD be used to represent parameter names that are not legal XML names.
- A non-void return value is represented as follows:
  1. There MUST be an outbound edge with a local name of `result` and a namespace name of "http://www.w3.org/2003/05/soap-rpc" which terminates in a terminal node

2. The type of that terminal node is a xs:QName and its value is the name of the outbound edge which terminates in the actual return value.

If the return value of the procedure is void then an outbound edge with a local name of `result` and a namespace name of "http://www.w3.org/2003/05/soap-rpc" MUST NOT be present.

- Invocation faults are handled according to the rules in 4.4 RPC Faults. If a protocol binding adds additional rules for fault expression, those MUST also be followed.

Comments

All tests in the test collection that use the RPC convention and do not generate a fault, test this assertion. Test 'T31' tests the void return case.

Assertion x2-rpcencrestriction-onechild

Location of the assertion

SOAP 1.2 Part 2, Section 4.2.3

Text from the specification

When using SOAP encoding (see 3. SOAP Encoding) in conjunction with the RPC convention described here, the SOAP `Body` MUST contain only a single child *element information item*, that child being the serialized RPC invocation or response struct.

Comments

All tests in the test collection that use the RPC convention, test this assertion.

Assertion x2-rpcsoaphead-hb

Location of the assertion

SOAP 1.2 Part 2, Section 4.3

Text from the specification

Additional information relevant to the encoding of an RPC invocation but not part of the formal procedure or method signature MAY be expressed in a SOAP envelope carrying an RPC invocation or response. Such additional information MUST be expressed as SOAP header blocks.

Tests

T32

Assertion x2-rpcfaults-rules

Location of the assertion

SOAP 1.2 Part 2, Section 4.4

Text from the specification

> Errors arising during RPC invocations are reported according to the following rules:
>
> 1. A fault with a `Value` of `Code` set to "env:Receiver" SHOULD be generated when the receiver cannot handle the message because of some temporary condition, e.g. when it is out of memory.
>
>   **Note:**
>
>   Throughout this document, the term "`Value` of `Code` " is used as a shorthand for "value of the `Value` child *element information item* of the `Code` *element information item*" (see SOAP 1.2 Part 1 [SOAP Part 1], SOAP Code Element).
> 2. A fault with a `Value` of `Code` set to "env:DataEncodingUnknown" SHOULD be generated when the arguments are encoded in a data encoding unknown to the receiver.
> 3. A fault with a `Value` of `Code` set to "env:Sender" and a `Value` of `Subcode` set to "rpc:ProcedureNotPresent" MAY be generated when the receiver does not support the procedure or method specified.
>
>   **Note:**
>
>   Throughout this document, the term "`Value` of `Subcode` " is used as a shorthand for "value of the `Value` child *element information item* of the `Subcode` *element information item*" (see SOAP 1.2 Part 1 [SOAP Part 1], SOAP Subcode element).
> 4. A fault with a `Value` of `Code` set to "env:Sender" and a `Value` of `Subcode` set to "rpc:BadArguments" MUST be generated when the receiver cannot parse the arguments or when there is a mismatch in number and/or type of the arguments between what the receiver expects and what was sent.
> 5. Other faults arising in an extension or from the application SHOULD be generated as described in SOAP 1.2 Part 1 [SOAP Part 1] SOAP Fault Codes.

Tests

T72, T69, T70, T33, T80

Assertion x2-bindprops-uri

Location of the assertion

SOAP 1.2 Part 2, Section 5.1.1

Text from the specification

> Properties are named with URIs.

Comments

This assertion will not be tested.

Assertion x2-meppropconv-def

Location of the assertion

SOAP 1.2 Part 2, Section 6.1

Text from the specification

> Table 2: Property definitions supporting the description of MEPs

Comments

This assertion will not be tested.

Assertion x2-mepname-uri

Location of the assertion

SOAP 1.2 Part 2, Section 6.2.1

Text from the specification

> This message exchange pattern is identified by the URI (see SOAP 1.2 Part 1 [SOAP Part 1] SOAP Features):
>
> - "http://www.w3.org/2003/05/soap/mep/request-response/"

Comments

This assertion will not be tested.

Assertion x2-bindformdesc-propdef

Location of the assertion

SOAP 1.2 Part 2, Section 6.2.3

Text from the specification

> Table 3: Property definitions for Request-Response MEP

Comments

This assertion will not be tested.

Assertion x2-bindformdesc-reqctx

Location of the assertion

SOAP 1.2 Part 2, Section 6.2.3

Text from the specification

> Table 4: Instantiation of a Message Exchange Context for a requesting SOAP node

Comments

This assertion will not be tested.

Assertion x2-bindformdesc-resctx

Location of the assertion

SOAP 1.2 Part 2, Section 6.2.3

Text from the specification

> Table 5: Instantiation of Message Exchange Context for an inbound request message at a responding SOAP node

Comments

This assertion will not be tested.

Assertion x2-bindformdesc-reqsttr

Location of the assertion

SOAP 1.2 Part 2, Section 6.2.3

Text from the specification

> Table 6: Requesting SOAP Node State Transitions

Comments

This assertion will not be tested.

Assertion x2-bindformdesc-ressttr

Location of the assertion

SOAP 1.2 Part 2, Section 6.2.3

Text from the specification

> Table 7: Responding SOAP Node State Transitions

Comments

This assertion will not be tested.

Assertion x2-bindformdesc-stream

Location of the assertion

SOAP 1.2 Part 2, Section 6.2.3

Text from the specification

> All the rules in SOAP 1.2 Part 1 [SOAP Part 1] Binding Framework regarding streaming of individual SOAP messages MUST be obeyed for both request and response SOAP messages.

Comments

This assertion will not be tested.

Assertion x2-bindformdesc-dlock

Location of the assertion

SOAP 1.2 Part 2, Section 6.2.3

Text from the specification

> When using streaming SOAP bindings, requesting SOAP nodes MUST avoid deadlock by accepting and if necessary processing SOAP response information while the SOAP request is being transmitted.

Comments

This assertion will not be tested.

Assertion x2-mepname2-uri

Location of the assertion

SOAP 1.2 Part 2, Section 6.3.1

Text from the specification

> This message exchange pattern is identified by the URI (see SOAP 1.2 Part 1 [SOAP Part 1] SOAP Features):
>
> - "http://www.w3.org/2003/05/soap/mep/soap-response/"

Comments

This assertion will not be tested.

Assertion x2-bindformdesc2-propdef

Location of the assertion

SOAP 1.2 Part 2, Section 6.3.3

Text from the specification

Table 8: Property definitions for SOAP Response MEP

Comments

This assertion will not be tested.

Assertion x2-bindformdesc2-reqctx

Location of the assertion

SOAP 1.2 Part 2, Section 6.3.3

Text from the specification

> Table 9: Instantiation of a Message Exchange Context for a requesting SOAP node

Comments

This assertion will not be tested.

Assertion x2-bindformdesc2-resctx

Location of the assertion

SOAP 1.2 Part 2, Section 6.3.3

Text from the specification

> Table 10: Instantiation of Message Exchange Context for an inbound request message

Comments

This assertion will not be tested.

Assertion x2-bindformdesc2-reqsttr

Location of the assertion

SOAP 1.2 Part 2, Section 6.3.3

Text from the specification

> Table 11: Requesting SOAP Node State Transitions

Comments

This assertion will not be tested.

Assertion x2-bindformdesc2-ressttr

Location of the assertion

SOAP 1.2 Part 2, Section 6.3.3

Text from the specification

> Table 12: Responding SOAP Node State Transitions

Comments

This assertion will not be tested.

Assertion x2-WebMethodFeatureName-uri

Location of the assertion

SOAP 1.2 Part 2, Section 6.4.1

Text from the specification

> The SOAP Web Method feature is identified by the URI (see SOAP 1.2 Part 1 [SOAP Part 1] SOAP Features):
>
> - "http://www.w3.org/2003/05/soap/features/web-method/"

Comments

This assertion will not be tested.

Assertion x2-webmethodstatemachine-urivalue

Location of the assertion

SOAP 1.2 Part 2, Section 6.4.3

Text from the specification

> A node sending a request message MUST provide a value for the `http://www.w3.org/2003/05/soap/features/web-method/Method` property.

Comments

This assertion will not be tested.

Assertion x2-webmethodstatemachine-compat

Location of the assertion

SOAP 1.2 Part 2, Section 6.4.3

Text from the specification

> Bindings implementing this feature MUST employee a Message Exchange Pattern with semantics that are compatible with the web method selected.

Comments

This assertion will not be tested.

Assertion x2-httpmediatype-mediatype

Location of the assertion

SOAP 1.2 Part 2, Section 7.1.4

Text from the specification

> Conforming implementations of this binding:
>
> 1. MUST be capable of sending and receiving messages serialized using media type "application/soap+xml" whose proper use and parameters are described in A. The application/soap+xml Media Type.

Comments

'this' in the specification text refer to the SOAP HTTP Binding.

Tests

TH1, TH2, TH3, TH4

Assertion x2-http-bindname-uri

Location of the assertion

SOAP 1.2 Part 2, Section 7.2

Text from the specification

> This binding is identified by the URI (see SOAP 1.2 Part 1 [SOAP Part 1] SOAP Protocol Binding Framework):
>
> - "http://www.w3.org/2003/05/soap/bindings/HTTP/"

Comments

This assertion will not be tested

Assertion x2-http-suptransmep-uris

Location of the assertion

SOAP 1.2 Part 2, Section 7.3

Text from the specification

> An implementation of the SOAP HTTP Binding MUST support the following message exchange patterns (MEPs):
>
> - "http://www.w3.org/2003/05/soap/mep/request-response/" (see 6.2 Request-Response Message Exchange Pattern)
> - "http://www.w3.org/2003/05/soap/mep/soap-response/" (see 6.3 SOAP Response Message Exchange Pattern)

Comments

This assertion will not be tested.

Assertion x2-http-suptfeatures-webmethod

Location of the assertion

SOAP 1.2 Part 2, Section 7.4

Text from the specification

> An implementation of the SOAP HTTP Binding MUST support the following feature:
>
> - "http://www.w3.org/2003/05/soap/features/web-method/" (see 6.4 Web Method Specification Feature)

Comments

This assertion will not be tested.

Assertion x2-http-suptfeatures-methrest

Location of the assertion

SOAP 1.2 Part 2, Section 7.4

Text from the specification

> The possible values of `http://www.w3.org/2003/05/soap/features/web-method/Method` property are restricted in this HTTP binding according to the MEP in use (as present in `http://www.w3.org/2003/05/soap/bindingFramework/ExchangeContext/ExchangePatternN`):
>
> Table 14: Possible values of the Web-Method Method property

Comments

This assertion will not be tested

Assertion x2-http-reqsoapnode-dlock

Location of the assertion

SOAP 1.2 Part 2, Section 7.5.1

Text from the specification

> This binding supports streaming and, as a result, requesting SOAP nodes MUST avoid deadlock by accepting and if necessary processing SOAP response information while the SOAP request is being transmitted (see 6.2.3 State Machine Description).

Comments

This assertion will not be tested.

Assertion x2-http-reqbindreqstate-reqfld

Location of the assertion

SOAP 1.2 Part 2, Section 7.5.1.1

Text from the specification

> Table 15: HTTP Request Fields

Tests

TH1, TH2, TH3, TH4

Assertion x2-http-reqbindwaitstate-trans

Location of the assertion

SOAP 1.2 Part 2, Section 7.5.1.2

Text from the specification

> Table 16: HTTP status code dependent transitions

Tests

TH1, TH2, TH3, TH4

Assertion x2-http-respbindreceive-initerror

Location of the assertion

SOAP 1.2 Part 2, Section 7.5.2.1

Text from the specification

> Table 17: Errors generated in the Init state

Tests

TH2, TH5

Assertion x2-http-respbindprocess-reshdr

Location of the assertion

SOAP 1.2 Part 2, Section 7.5.2.2

Text from the specification

> Table 18: HTTP Response Headers Fields

Tests

TH1, TH2, TH3, TH4, TH5

Assertion x2-http-respbindprocess-fltmap

Location of the assertion

SOAP 1.2 Part 2, Section 7.5.2.2

Text from the specification

> Table 19: SOAP Fault to HTTP Status Mapping

Tests

TH2, TH3, TH4, TH5

Assertion x2-namemap-rules-sql

Location of the assertion

SOAP 1.2 Part 2, Appendix B.1

Text from the specification

> Rules for Mapping Application Defined Names to XML Names

Comments

This assertion will not be tested.

3. SOAP 1.2 Test Collection

3.1 Introduction

Unless otherwise stated all the tests in this test collection follow the following rules:

- The tests use three SOAP nodes - Node A, Node B and Node C, identified by "http://example.org/ts-tests/A", "http://example.org/ts-tests/B", and "http://example.org/ts-tests/C" respectively. No other SOAP nodes must be used in communication between these three SOAP nodes.
- Node A is the test client.
- Node C is the ultimate destination.
- Node B is a SOAP intermediary.
- Node B must act in the role "http://example.org/ts-tests/B"
- Node C must act in the role "http://example.org/ts-tests/C"
- Node A, Node B and Node C implement some mechanism for routing so that the following messaging scenarios are allowed:
    - Node A sends message to Node C, Node C returns a response or fault message back to Node A (Node B is not involved in this scenario).
    - Node A sends message to Node B, Node B forwards the message to Node C or returns a fault back to Node A. Node C either:
        - returns a fault message to Node B and Node B forwards the fault message to Node A, OR
        - returns a response message to Node B and Node B forwards the response to Node A.

3.2 Header Blocks Used by the Test Collection

Unless otherwise specified the header blocks used by this test collection are in the namespace "http://example.org/ts-tests".

3.2.1 echoOk

The semantics of processing this header block require the SOAP node targeted by this header block, to reply to the SOAP node from which it received the SOAP message containing this header. The reponse SOAP message must contain the header block *responseOk* containing the same information set as that in *echoOk* header block. The type of this header block is string in the namespace "http://www.w3.org/2001/XMLSchema".

3.2.2 responseOk

This header block is generated as a result of processing the *echoOk* header block as described above. The type of this header block is string in the namespace "http://www.w3.org/2001/XMLSchema".

3.2.3 Ignore

The semantics of processing this header block require the SOAP node targeted by this header block, to ignore this header block altogether. The type of this header block is string in the namespace "http://www.w3.org/2001/XMLSchema".

3.2.4 requiredHeader

This header block is used in conjunction with the body block *echoHeader* The semantics of processing the body block *echoHeader* requires the SOAP node to reply to the SOAP node from which it received the SOAP message containing this header. The response SOAP message must contain the SOAP body block *echoHeaderResponse* containing the same information set as that in *requiredHeader* header block. The type of this header block is string in the namespace "http://www.w3.org/2001/XMLSchema".

3.2.5 DataHolder

The semantics of processing this header block require the SOAP node targeted by this header block, to ignore this header block altogether. This header is used for encapsulating data used by other headers and body blocks. The type of this header block is string in the namespace "http://www.w3.org/2001/XMLSchema".

3.2.6 concatAndForwardEchoOk

The semantics of processing this header block require the SOAP node targeted by this header block, to take the character information item children of the header block *concatAndForwardEchoOkArg2* concatenate it to the character information item children of the header block *concatAndForwardEchoOkArg1* and forward the result to the downstream SOAP node using the header block *echoOK* This header should not contain any character information item children.

3.2.7 concatAndForwardEchoOkArg1

The semantics of processing this header block require the SOAP node targeted by this header block, to ignore this header block altogether. This header is used for encapsulating data used by the header *concatAndForwardEchoOk* block. The type of this header block is string in the namespace "http://www.w3.org/2001/XMLSchema".

3.2.8 concatAndForwardEchoOkArg2

The semantics of processing this header block require the SOAP node targeted by this header block, to ignore this header block altogether. This header is used for encapsulating data used by the header *concatAndForwardEchoOk* block. The type of this header block is string in the namespace "http://www.w3.org/2001/XMLSchema".

3.2.9 validateCountryCode

The semantics of processing this header block require the SOAP node targeted by this header block, to validate that the character information item of this header consists of two letters only (ignoring whitespace). If this condition is not satisfied then a fault is required to be sent back to the sender of the message with the Value of the fault Code as env:Sender along with a header block *validateCountryCodeFault* containing an explanation for the fault The type of this header block is string in the namespace "http://www.w3.org/2001/XMLSchema".

3.2.10 validateCountryCodeFault

This header block is used to cary information related to fault generated as a result of processing the header block *validateCountryCode* as described above. The type of this header block is string in the namespace "http://www.w3.org/2001/XMLSchema".

3.2.11 echoResolvedRef

The semantics of processing this header block require the SOAP node targeted by this header block, to reply to the SOAP node from which it received the SOAP message containing this header. The reponse SOAP message must contain the header block *responseResolvedRef* This header block contains one child element information item RelativeReference in the namespace "http://example.org/ts-tests". RelativeReference element information item is required to have the attribute information items xml:base, and xlink:href. The *responseResolvedRef* contains the resolved reference pointed to by xml:base and xlink:href.

3.2.12 responseResolvedRef

This header block is generated in response to processing the header block *echoResolvedRef* as described above. The type of this header block is string in the namespace "http://www.w3.org/2001/XMLSchema".

3.3 Body Blocks Used by the Test Collection

Unless otherwise specified the body blocks used by this test collection are in the namespace "http://example.org/ts-tests".

3.3.1 echoOk

The semantics of processing this body block require the SOAP node to reply to the SOAP node from which it received the SOAP message containing this block. The reponse SOAP message must contain the body block *responseOk* containing the same information set as that in *echoOk* body block. The type of this header block is string in the namespace "http://www.w3.org/2001/XMLSchema".

3.3.2 responseOk

This body block is generated as a result of processing the *echoOk* body block as described above. The type of this header block is string in the namespace "http://www.w3.org/2001/XMLSchema".

3.3.3 echoHeader

This body block is used in conjuction with the header block *requiredHeader* The semantics of processing this body block require the SOAP node to reply to the SOAP node from which it received the SOAP message containing this block. The response SOAP message must contain the body block *echoHeaderResponse* containing the same information set as that in *requiredHeader* header block. This body block does not have any children element information items, or attribute information items.

3.3.4 echoHeaderResponse

This body block is generated as a result of processing the *echoHeader* body block as described above. The type of this header block is string in the namespace "http://www.w3.org/2001/XMLSchema".

3.4 RPC Methods/Procedures Used by the Test Collection

Unless otherwise specified the procedure/method names used by this test collection are in the namespace "http://example.org/ts-tests".

In addition to types defined in the namespace "http://www.w3.org/2001/XMLSchema", the test collection uses the following types:

- `SOAPStruct` defined in the namespace "http://example.org/ts-tests/xsd". This type contains three child element information items in its children property as follows:
    - An element information item of type string in the namespace "http://www.w3.org/2001/XMLSchema".
    - An element information item of type int in the namespace "http://www.w3.org/2001/XMLSchema".
    - An element information item of type float in the namespace "http://www.w3.org/2001/XMLSchema".
- `SOAPStructStruct` defined in the namespace "http://example.org/ts-tests/xsd". This type contains four child element information items in its children property as follows:
    - An element information item of type int in the namespace "http://www.w3.org/2001/XMLSchema".
    - An element information item of type float in the namespace "http://www.w3.org/2001/XMLSchema".
    - An element information item of type string in the namespace "http://www.w3.org/2001/XMLSchema".
    - An element information item of type SOAPStruct in the namespace

"http://example.org/ts-tests/xsd".

- `SOAPArrayStruct` defined in the namespace "http://example.org/ts-tests/xsd". This type contains four child element information items in its children property as follows:
  - An element information item of type int in the namespace "http://www.w3.org/2001/XMLSchema".
  - An element information item of type float in the namespace "http://www.w3.org/2001/XMLSchema".
  - An element information item of type string in the namespace "http://www.w3.org/2001/XMLSchema".
  - An element information item representing an array of type string in the namespace "http://www.w3.org/2001/XMLSchema".

The encoding represented by the URI "http://example.org/PoisonEncoding" is an encoding that is not recognized by any of the SOAP nodes.

Some of the tests in this test collection, test SOAP 1.2 HTTP binding. The request and response messages for these tests contain HTTP start-line (request-line or status-line), HTTP headers required by the bindings and the XML payload. Additional HTTP headers can be generated in accordance with the rules for the binding specific expression of any optional features in use for this message exchange. In the tests, the value of the 'Content-Length' and 'Host' header should be replaced with an appropriate value.

3.4.1 returnVoid

This procedure/method does not have any input and output parameters and does not have a return value.

3.4.2 echoStruct

This procedure/method has one input parameter and a return value. Both are of type SOAPStruct in the name space "http://example.org/ts-tests/xsd". The semantics of this method consists of returning the input argument in the response.

3.4.3 echoStructArray

This procedure/method has one input parameter and a return value. Both are of type array of SOAPStruct in the name space "http://example.org/ts-tests/xsd". The semantics of this method consists of returning the input argument in the response.

3.4.4 echoStructAsSimpleTypes

This procedure/method has one input parameter and three return value. The input parameter is of type SOAPStruct in the name space "http://example.org/ts-tests/xsd". The first output parameter is of type int in the namespace "http://www.w3.org/2001/XMLSchema". The second output parameter is of type float in the namespace "http://www.w3.org/2001/XMLSchema". The third output parameter is of type string in the namespace "http://www.w3.org/2001/XMLSchema". The semantics of this method consists of returning the individual members of SOAPStruct in the input argument as output arguments, in the response.

3.4.5 echoSimpleTypesAsStruct

This procedure/method has three input parameter and a return value. The first input parameter is of type int in the namespace "http://www.w3.org/2001/XMLSchema". The second input parameter is of type float in the namespace "http://www.w3.org/2001/XMLSchema". The third input parameter is of type string in the namespace "http://www.w3.org/2001/XMLSchema". The return type is SOAPStruct in the name space "http://example.org/ts-tests/xsd". The semantics of this method consists of using the input arguments to construct an instance of SOAPStruct and returning the result in the response.

3.4.6 echoNestedStruct

This procedure/method has one input parameter and a return value. Both are of type SOAPStructStruct in the name space "http://example.org/ts-tests/xsd". The semantics of this method consists of returning the input argument in the response.

3.4.7 echoNestedArray

This procedure/method has one input parameter and a return value. Both are of type SOAPArrayStruct in the name space "http://example.org/ts-tests/xsd". The semantics of this method consists of returning the input argument in the response.

3.4.8 echoFloatArray

This procedure/method has one input parameter and a return value. Both are of type array of float in the name space "http://www.w3.org/2001/XMLSchema". The semantics of this method consists of returning the input argument in the response.

3.4.9 echoStringArray

This procedure/method has one input parameter and a return value. Both are of type array of string in the name space "http://www.w3.org/2001/XMLSchema". The semantics of this method consists of returning the input argument in the response.

3.4.10 echoIntegerArray

This procedure/method has one input parameter and a return value. Both are of type array of int in the name space "http://www.w3.org/2001/XMLSchema". The semantics of this method consists of returning the input argument in the response.

3.4.11 echoBase64

This procedure/method has one input parameter and a return value. Both are of type base64Binary in the name space "http://www.w3.org/2001/XMLSchema". The semantics of this method consists of returning the input argument in the response.

3.4.12 echoBoolean

This procedure/method has one input parameter and a return value. Both are of type boolean in the name space "http://www.w3.org/2001/XMLSchema". The semantics of this method consists of returning the input argument in the response.

3.4.13 echoDate

This procedure/method has one input parameter and a return value. Both are of type date in the name space "http://www.w3.org/2001/XMLSchema". The semantics of this method consists of returning the input argument in the response.

3.4.14 echoDecimal

This procedure/method has one input parameter and a return value. Both are of type decimal in the name space "http://www.w3.org/2001/XMLSchema". The semantics of this method consists of returning the input argument in the response.

3.4.15 echoFloat

This procedure/method has one input parameter and a return value. Both are of type float in the name space "http://www.w3.org/2001/XMLSchema". The semantics of this method consists of returning the input argument in the response.

3.4.16 echoString

This procedure/method has one input parameter and a return value. Both are of type string in the name space "http://www.w3.org/2001/XMLSchema". The semantics of this method consists of returning the input argument in the response.

3.4.17 countItems

This procedure/method has one input parameter and a return value. The input parameter is of type array of string in the name space "http://www.w3.org/2001/XMLSchema" and the type of the return value is int in the namespace "http://www.w3.org/2001/XMLSchema". The semantics of this method consists of returning the cardinality of the input array argument in the response.

3.4.18 isNil

This procedure/method has one input parameter and a return value. The input parameter is of type string in the name space "http://www.w3.org/2001/XMLSchema" and the type of the return value is boolean in the namespace "http://www.w3.org/2001/XMLSchema". The semantics of this method consists of returning a boolean 'true', if the input argument is absent or has xsi:nil attribute with a value of '1'. A boolean 'false' is returned otherwise.

3.5 Tests

Test:T1

Description:

Node A sends to Node C message with 'echoOk' header block having role equal to "http://www.w3.org/2003/05/soap-envelope/role/next". Node C returns back empty body with responseOK header.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:responseOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:responseOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Test:T2

Description:

Node A sends to Node C message with echoOk header block having role equal to "http://example.org/ts-tests/C". NodeC returns back empty body with responseOK header.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
```

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://example.org/ts-tests/C">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <test:responseOk xmlns:test="http://example.org/ts-tests">
    foo
  </test:responseOk>
 </env:Header>
 <env:Body>
 </env:Body>
</env:Envelope>
```

Test:T3

Description:

Node A sends to Node C message with echoOk header block having no role. NodeC returns back empty body with responseOK header.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:responseOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:responseOk>
  </env:Header>
```

```
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Test:T4

Description:

Node A sends to node C message with echoOk header block having role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver". NodeC sends message back with responseOK header.

Messages:

Message sent from Node A

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:echoOk xmlns:test="http://example.org/ts-tests"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
        foo
      </test:echoOk>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:responseOk xmlns:test="http://example.org/ts-tests">
        foo
      </test:responseOk>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Test:T5

Description:

Node A sends to node C message with echoOk header block having role="http://example.org/ts-tests/B". Node C sends response message without a responseOK header and an empty body (header block was ignored).

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://example.org/ts-tests/B">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
  </env:Body>
</env:Envelope>
```

Test:T6

Description:

Node A sends to node B message with echoOk header block having role="http://example.org/ts-tests/C". NodeB forwards message to NodeC without touching the header block.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://example.org/ts-tests/C">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node B

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://example.org/ts-tests/C">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <test:responseOk xmlns:test="http://example.org/ts-tests">
    foo
  </test:responseOk>
 </env:Header>
 <env:Body>
 </env:Body>
</env:Envelope>
```

Test:T7

Description:

Node A sends to node B message with Unknown header block having role="http://example.org/ts-tests/B". NodeB forwards message to NodeC with no header (header was removed).

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:Ignore xmlns:test="http://example.org/ts-tests"
          env:role="http://example.org/ts-tests/B">
      foo
    </test:Ignore>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node B

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
```

```
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Test:T8

Description:

Node A sends to node B message with 3 headers: echoOk header block having no role, Ignore header block having role="http://example.org/ts-tests/B" and echoOk header block having role="http://www.w3.org/2003/05/soap-envelope/role/none". NodeB removes the second header block, that has role="http://example.org/ts-tests/B" and forwards message to NodeC with 2 other headers included in the same order as in the original message.

Messages:

Message sent from Node A

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:echoOk xmlns:test="http://example.org/ts-tests">
        foo
      </test:echoOk>
      <test:Ignore xmlns:test="http://example.org/ts-tests"
            env:role="http://example.org/ts-tests/B">
        foo
      </test:Ignore>
      <test:echoOk xmlns:test="http://example.org/ts-tests"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/none">
        foo
      </test:echoOk>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Message sent from Node B

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:echoOk xmlns:test="http://example.org/ts-tests">
        foo
      </test:echoOk>
      <test:echoOk xmlns:test="http://example.org/ts-tests"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/none">
        foo
      </test:echoOk>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:responseOk xmlns:test="http://example.org/ts-tests">
        foo
      </test:responseOk>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Test:T9

Description:

Node A sends to node B message with echoOk header block having role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver". NodeB forwards message to NodeC with header included.

Messages:

Message sent from Node A

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:echoOk xmlns:test="http://example.org/ts-tests"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
        foo
      </test:echoOk>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Message sent from Node B

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:echoOk xmlns:test="http://example.org/ts-tests"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
        foo
      </test:echoOk>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:responseOk xmlns:test="http://example.org/ts-tests">
        foo
      </test:responseOk>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Test:T10

Description:

Node A sends to node C message with Unknown header having role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver". NodeC sends empty message back with no headers - header is ignored.

Messages:

Message sent from Node A

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:Unknown xmlns:test="http://example.org/ts-tests"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
        foo
      </test:Unknown>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Test:T11

Description:

Node A sends to node C message with Unknown header with mustUnderstand="false" and having role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver". NodeC sends empty message back with no headers - header is ignored.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:Unknown xmlns:test="http://example.org/ts-tests"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
          env:mustUnderstand="false">
      foo
    </test:Unknown>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
  </env:Body>
</env:Envelope>
```

Test:T12

Description:

Node A sends to node C message with Unknown header with mustUnderstand="1" and having role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver". NodeC sends MustUnderstand fault back.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
```

```
      <test:Unknown xmlns:test="http://example.org/ts-tests"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
            env:mustUnderstand="1">
        foo
      </test:Unknown>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <env:NotUnderstood qname="test:Unknown"
         xmlns:test="http://example.org/ts-tests" />
    </env:Header>
    <env:Body>
      <env:Fault>
        <env:Code>
          <env:Value>env:MustUnderstand</env:Value>
        </env:Code>
        <env:Reason>
          <env:Text xml:lang="en-US"> Header not understood </env:Text>
        </env:Reason>
      </env:Fault>
    </env:Body>
  </env:Envelope>
```

Test:T13

Description:

Node A sends to node C message with Unknown header with mustUnderstand="true" and having role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver". NodeC sends MustUnderstand fault back.

Messages:

Message sent from Node A

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:Unknown xmlns:test="http://example.org/ts-tests"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
            env:mustUnderstand="true">
        foo
      </test:Unknown>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <env:NotUnderstood qname="test:Unknown"
       xmlns:test="http://example.org/ts-tests" />
  </env:Header>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:MustUnderstand</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US">
          Header not understood
        </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T14

Description:

Node A sends to node C message with echoOk header with mustUnderstand="wrong" and having role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver". NodeC sends a fault back.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
          env:mustUnderstand="wrong">
      foo
    </test:echoOK>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
       </env:Code>
       <env:Reason>
        <env:Text xml:lang="en-US">
          env:mustUnderstand value is not boolean
        </env:Text>
       </env:Reason>
     </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T15

Description:

Node A sends to node C message with Unknown header with mustUnderstand="1" and having role="http://example.org/ts-tests/B". NodeC sends empty message back with no headers- header is ignored.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:Unknown xmlns:test="http://example.org/ts-tests"
          env:role="http://example.org/ts-tests/B"
          env:mustUnderstand="1">
      foo
    </test:Unknown>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
  </env:Body>
</env:Envelope>
```

Test:T16

Description:

Node A sends to node B message with Unknown header with mustUnderstand="1" and having role="http://example.org/ts-tests/C". NodeB forwards message to node C keeping header untouched.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:Unknown xmlns:test="http://example.org/ts-tests"
          env:role="http://example.org/ts-tests/C"
          env:mustUnderstand="1">
      foo
    </test:Unknown>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <env:NotUnderstood qname="test:Unknown"
       xmlns:test="http://example.org/ts-tests" />
  </env:Header>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:MustUnderstand</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US"> Header not understood </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T17

Description:

Node A sends to node B message with Unknown header with mustUnderstand="1" and having role="http://www.w3.org/2003/05/soap-envelope/role/next". NodeB returns MustUnderstand fault to node A.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
```

```
  <env:Header>
    <test:Unknown xmlns:test="http://example.org/ts-tests"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="1">
      foo
    </test:Unknown>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node B

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <env:NotUnderstood qname="test:Unknown"
       xmlns:test="http://example.org/ts-tests"/>
  </env:Header>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:MustUnderstand</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US"> Header not understood </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T18

Description:

Node A sends to node B message with echoOk header having role="http://www.w3.org/2003/05/soap-envelope/role/none". NodeB forwards message to the node C, node C responds back to node A with empty message (no body/header blocks).

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/none">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node B

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/none">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
  </env:Body>
</env:Envelope>
```

Test:T19

Description:

Node A sends to node C message with echoOk header having mustUnderstand = "true" and role="http://www.w3.org/2003/05/soap-envelope/role/none". Node C ignores this header block and returns empty message (no body/header blocks).

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/none"
          env:mustUnderstand="true">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
  </env:Body>
</env:Envelope>
```

Test:T21

Description:

Node A sends to node B message with Unknown header that has mustUnderstand = "1" and role="http://example.org/ts-tests/B" and echoOk header that has role="http://example.org/ts-tests/C" role and mustUnderstand="1". Node B must return MustUnderstand Fault message to Node A and no message should be forwarded to Node C.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
```

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:Unknown xmlns:test="http://example.org/ts-tests"
          env:mustUnderstand="1"
          env:role="http://example.org/ts-tests/B">
      foo
    </test:Unknown>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:mustUnderstand="1"
          env:role="http://example.org/ts-tests/C">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node B

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <env:NotUnderstood qname="test:Unknown"
       xmlns:test="http://example.org/ts-tests" />
  </env:Header>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:MustUnderstand</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US"> Header not understood </env:Text>
      </env:Reason>
      <env:Node>http://example.org/ts-tests/B</env:Node>
      <env:Role>http://example.org/ts-tests/B</env:Role>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T22

Description:

Node A sends to Node C message with echoOk header that has mustUnderstand="1" and echoOk Body element. NodeC must process the Header and the Body and return to Node A message with responseOk header and responseOk Body element.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:mustUnderstand = "1">
```

```
        foo
      </test:echoOk>
    </env:Header>
    <env:Body>
      <test:echoOk xmlns:test="http://example.org/ts-tests">
        foo
      </test:echoOk>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:responseOk xmlns:test="http://example.org/ts-tests">
        foo
      </test:responseOk>
    </env:Header>
    <env:Body>
      <test:responseOk xmlns:test="http://example.org/ts-tests">
        foo
      </test:responseOk>
    </env:Body>
  </env:Envelope>
```

Test:T23

Description:

Node A sends to Node C message with echoOk header that has mustUnderstand="wrong" and Unknown header that has mustUnderstand="1". Node C should return exactly one fault.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:Unknown xmlns:test="http://example.org/ts-tests"
          env:mustUnderstand="1">
      foo
    </test:Unknown>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:mustUnderstand="wrong">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
       </env:Code>
       <env:Reason>
        <env:Text xml:lang="en-US">
          env:mustUnderstand value is not boolean
        </env:Text>
       </env:Reason>
     </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T24

Description:

Node A sends to node C message with incorrect namespace of the Envelope element.

Node C returns back VersionMismatch Fault.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://example.org/wrong-version/">
  <env:Body>
    <test:echoOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:echoOk>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:VersionMismatch</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US"> Wrong Version </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T25

Description:

Node A sends to node C message with reference to external DTD. Node C returns back DTDNotSupported Fault.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<!DOCTYPE env:Envelope SYSTEM "env.dtd"[]>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <test:echoOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:echoOk>
 </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
     <env:Fault>
       <env:Code>
        <env:Value>env:Sender</env:Value>
       </env:Code>
       <env:Reason>
        <env:Text xml:lang="en-US">
          DTD are not supported by SOAP 1.2
        </env:Text>
       </env:Reason>
     </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T26

Description:

Node A sends to node C message with Processing Instruction node. Node C ignores PI and returns back Body with test:responseOk element.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
<?xml-stylesheet href="http://example.org/ts-tests/sub.xsl" type = "text/xsl"?>
  <env:Body>
    <test:echoOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:echoOk>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <test:responseOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:responseOk>
 </env:Body>
</env:Envelope>
```

Test:T27

Description:

Node A sends to node C message with test:echoStringArray that has encodingStyle attribute with a value of "http://www.w3.org/2003/05/soap-encoding", contains an element with attribute enc:itemType="xsd:string" (array of string), but with the child element of a complex type. Node C returns a Fault indicating that message didn't follow SOAP encoding rules (encoded array content didn't correspond to the type declared in the enc:itemType).

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <test:echoStringArray xmlns:test="http://example.org/ts-tests"
          xmlns:enc="http://www.w3.org/2003/05/soap-encoding"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <test:array enc:itemType="xs:string" enc:arraySize="1">
        <a>
          <b>1</b>
        </a>
      </test:array>
    </test:echoStringArray>
 </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
         <env:Subcode>
           <env:Value xmlns:rpc="http://www.w3.org/2003/05/soap-rpc">
             rpc:BadArguments
           </env:Value>
         </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US">
          Violation of encoding rules
        </env:Text>
      </env:Reason>
      <env:Detail>
        Array element declared as array of string contains elements
        with wrong content.
      </env:Detail>
     </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T28

Description:

Node A sends to node C message with Body element that has encodingStyle attribute. Node C returns Fault message, because Body element must not contain attributes.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
    <test:echoOk xmlns:test="http://example.org/ts-tests" >
      foo
    </test:echoOk>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US">
          Incorrect SOAP Body element serialization
        </env:Text>
      </env:Reason>
      <env:Detail>
        SOAP Body must not have encodingStyle attribute information item.
      </env:Detail>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T29

Description:

Node A sends to node C message with Header having a role attribute value (URI) of length 2K role. Node C returns back empty body.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://example.org/ts-tests/Czzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
  </env:Body>
</env:Envelope>
```

Test:T30

Description:

A SOAP 1.1 sender, Node A, sends a 1.1 message to a SOAP Version 1.2 Node C. Node C may return back a VersionMismatch fault (tested here) or process the message (if it supports SOAP 1.1).

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope">
  <env:Body>
    <test:echoOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:echoOk>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope">
  <env:Header>
```

```
      <env:Upgrade>
        <env:SupportedEnvelope qname="ns2:Envelope"
                  xmlns:ns2="http://www.w3.org/2003/05/soap-envelope"/>
       </env:Upgrade>
    </env:Header>
    <env:Body>
      <env:Fault>
        <faultcode>env:VersionMismatch</faultcode>
        <faultstring>Wrong Version</faultstring>
      </env:Fault>
    </env:Body>
  </env:Envelope>
```

Test:T31

Description:

Node A sends to Node C an RPC message. Node C returns a void return value. Note that the return value accessor MUST NOT be present.

Messages:

Message sent from Node A

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
      <test:returnVoid xmlns:test="http://example.org/ts-tests">
      </test:returnVoid>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
      <test:returnVoidResponse xmlns:test="http://example.org/ts-tests">
      </test:returnVoidResponse>
   </env:Body>
  </env:Envelope>
```

Test:T32

Description:

Node A sends to Node C an RPC message with a required header. Node C returns the value supplied in the header.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:requiredHeader xmlns:test="http://example.org/ts-tests"
          env:mustUnderstand="true">
      foo
    </test:requiredHeader>
  </env:Header>
  <env:Body>
    <test:echoHeader xmlns:test="http://example.org/ts-tests">
    </test:echoHeader>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <test:echoHeaderResponse xmlns:test="http://example.org/ts-tests">
      foo
    </test:echoHeaderResponse>
  </env:Body>
</env:Envelope>
```

Test:T33

Description:

Node A sends to node C an RPC message with a procedure it cannot find. Node C returns an fault with env:Sender as the Value for the fault and rpc:ProcedureNotPresent as the Value for the SubCode. Please note that the Value of rpc:ProcedureNotPresent for the SubCode is not required by SOAP 1.2

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <test:DoesNotExist xmlns:test="http://example.org/ts-tests">
  </test:DoesNotExist>
 </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:rpc="http://www.w3.org/2003/05/soap-rpc">
  <env:Body>
     <env:Fault>
       <env:Code>
         <env:Value>env:Sender</env:Value>
         <env:Subcode>
           <env:Value>rpc:ProcedureNotPresent</env:Value>
         </env:Subcode>
       </env:Code>
       <env:Reason>
        <env:Text xml:lang="en-US"> Procedure Not Present </env:Text>
       </env:Reason>
     </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T34

Description:

This test consists of Node A sending a msg with a header that has MU=1 in the SOAP 1.1 NS. Node C, ignores this header.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:Unknown xmlns:test="http://example.org/ts-tests"
          xmlns:env1="http://schemas.xmlsoap.org/soap/envelope/"
          env1:mustUnderstand="true">
      foo
    </test:Unknown>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
  </env:Body>
</env:Envelope>
```

Test:T35

Description:

This test consists of Node A sending a msg with a Unknown header with MU=1 and role not specified. Node C returns a fault.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:Unknown xmlns:test="http://example.org/ts-tests"
          env:mustUnderstand="1">
      foo
    </test:Unknown>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <env:NotUnderstood qname="test:Unknown"
       xmlns:test="http://example.org/ts-tests" />
  </env:Header>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:MustUnderstand</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US"> Header not understood </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T36

Description:

This test consists of Node A sending a msg with a unknown header that is targeted to the ultimate receiver and has the role attribute set to http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver Node C returns a fault

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:Unknown xmlns:test="http://example.org/ts-tests"
          env:mustUnderstand="1"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
      foo
    </test:Unknown>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <env:NotUnderstood qname="test:Unknown"
       xmlns:test="http://example.org/ts-tests" />
  </env:Header>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:MustUnderstand</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US"> Header not understood </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
  </env:Envelope>
```

Test:T37

Description:

This test consists of Node A sending a msg with a header that does not have MU attr defined. Node C returns a valid reply.

Messages:

Message sent from Node A

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:Unknown xmlns:test="http://example.org/ts-tests"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver">
        foo
      </test:Unknown>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Test:T38

Description:

This test consists of Node A sending a msg with a header that has MU attr with all possible lexical values.

First set of messages.

Second set of messages.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:Unknown xmlns:test="http://example.org/ts-tests"
          env:mustUnderstand="false"
          env:role="http://example.org/ts-tests/C">
      foo
    </test:Unknown>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:mustUnderstand="0"
          env:role="http://example.org/ts-tests/C">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:responseOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:responseOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:mustUnderstand="true"
          env:role="http://example.org/ts-tests/C">
      foo
    </test:echoOk>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
```

```
            env:mustUnderstand="1"
            env:role="http://example.org/ts-tests/C">
        bar
      </test:echoOk>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:responseOk xmlns:test="http://example.org/ts-tests">
        foo
      </test:responseOk>
      <test:responseOk xmlns:test="http://example.org/ts-tests">
        bar
      </test:responseOk>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Test:T39

Description:

This test consists of Node A sending a msg with an unknown header and an incorrect value for the mustUnderstand attribute. Node C returns a env:Sender fault.

Messages:

Message sent from Node A

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:Unknown xmlns:test="http://example.org/ts-tests"
            env:mustUnderstand="9">
        foo
      </test:Unknown>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
      <env:Fault>
        <env:Code>
```

```
          <env:Value>env:Sender</env:Value>
        </env:Code>
        <env:Reason>
          <env:Text xml:lang="en-US">
            env:mustUnderstand is a xsd:boolean
          </env:Text>
        </env:Reason>
      </env:Fault>
    </env:Body>
  </env:Envelope>
```

Test:T40

Description:

This test uses the literal format for IPv6 addresses in URIs.

Messages:

Message sent from Node A

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:Unknown xmlns:test="http://[FEDC:BA98:7654:3210:FEDC:BA98:7654:3210]/ts-
            env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
            env:mustUnderstand="false">
        foo
      </test:Unknown>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Test:T41

Description:

Node A sends a message to Node C contained in echoStruct with datatype SOAPStruct in the namespace "http://example.org/ts-tests/xsd". Node C returns the datatypes in echoStructResponse.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStruct xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputStruct xsi:type="ns1:SOAPStruct"
                   xmlns:ns1="http://example.org/ts-tests/xsd">
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
        <varString xsi:type="xsd:string">hello world</varString>
      </inputStruct>
    </test:echoStruct>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStructResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="ns1:SOAPStruct"
              xmlns:ns1="http://example.org/ts-tests/xsd">
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
        <varString xsi:type="xsd:string">hello world</varString>
      </return>
    </test:echoStructResponse>
  </env:Body>
</env:Envelope>
```

Test:T42

Description:

Node A sends a message to Node C in echoStructArray containing an element with attribute enc:itemType="http://example.org/ts-tests/xsd:SOAPStruct". Node C responds with echoStructArrayResponse.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStructArray xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputStructArray enc:itemType="ns1:SOAPStruct"
                        enc:arraySize="2"
                        xmlns:ns1="http://example.org/ts-tests/xsd"
                        xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="ns1:SOAPStruct">
          <varInt xsi:type="xsd:int">42</varInt>
          <varFloat xsi:type="xsd:float">0.005</varFloat>
          <varString xsi:type="xsd:string">hello world</varString>
        </item>
        <item xsi:type="ns1:SOAPStruct">
          <varInt xsi:type="xsd:int">43</varInt>
          <varFloat xsi:type="xsd:float">0.123</varFloat>
          <varString xsi:type="xsd:string">bye world</varString>
        </item>
      </inputStructArray>
    </test:echoStructArray>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStructArrayResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return enc:itemType="ns1:SOAPStruct"
              enc:arraySize="2"
              xmlns:ns1="http://example.org/ts-tests/xsd"
              xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="ns1:SOAPStruct">
          <varInt xsi:type="xsd:int">42</varInt>
          <varFloat xsi:type="xsd:float">0.005</varFloat>
          <varString xsi:type="xsd:string">hello world</varString>
        </item>
        <item xsi:type="ns1:SOAPStruct">
          <varInt xsi:type="xsd:int">43</varInt>
          <varFloat xsi:type="xsd:float">0.123</varFloat>
          <varString xsi:type="xsd:string">bye world</varString>
        </item>
      </return>
    </test:echoStructArrayResponse>
  </env:Body>
</env:Envelope>
```

Test:T43

Description:

Node A sends a message to Node C in echoStructAsSimpleTypes containing datatypes SOAPStruct in the namespace "http://example.org/ts-tests/xsd" Node C responds with echoStructAsSimpleTypesResponse.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStructAsSimpleTypes xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputStruct xsi:type="ns1:SOAPStruct"
                   xmlns:ns1="http://example.org/ts-tests/xsd">
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
        <varString xsi:type="xsd:string">hello world</varString>
      </inputStruct>
    </test:echoStructAsSimpleTypes>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStructAsSimpleTypesResponse xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <outputInt xsi:type="xsd:int">42</outputInt>
      <outputFloat xsi:type="xsd:float">0.005</outputFloat>
      <outputString xsi:type="xsd:string">hello world</outputString>
    </test:echoStructAsSimpleTypesResponse>
  </env:Body>
</env:Envelope>
```

Test:T44

Description:

Node A sends a message to Node C in echoSimpleTypesAsStruct containing datatypes (integers, floating point, and string). Node C responds with echoSimpleTypesAsStructResponse.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoSimpleTypesAsStruct xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputInt xsi:type="xsd:int">42</inputInt>
      <inputFloat xsi:type="xsd:float">0.005</inputFloat>
      <inputString xsi:type="xsd:string">hello world</inputString>
    </test:echoSimpleTypesAsStruct>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoSimpleTypesAsStructResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="ns1:SOAPStruct"
              xmlns:ns1="http://example.org/ts-tests/xsd">
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
        <varString xsi:type="xsd:string">hello world</varString>
      </return>
    </test:echoSimpleTypesAsStructResponse>
  </env:Body>
</env:Envelope>
```

Test:T45

Description:

Node A sends a message to Node C in echoNestedStruct containing datatypes (integers, floating point, and string). Node C responds with echoNestedStruct.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoNestedStruct xmlns:test="http://example.org/ts-tests"
       env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputStruct xsi:type="ns1:SOAPStructStruct"
                   xmlns:ns1="http://example.org/ts-tests/xsd">
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
        <varString xsi:type="xsd:string">hello world</varString>
        <varStruct xsi:type="ns1:SOAPStruct">
          <varInt xsi:type="xsd:int">99</varInt>
          <varFloat xsi:type="xsd:float">4.0699e-12</varFloat>
          <varString xsi:type="xsd:string">nested struct</varString>
        </varStruct>
      </inputStruct>
    </test:echoNestedStruct>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoNestedStructResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="ns1:SOAPStructStruct"
              xmlns:ns1="http://example.org/ts-tests/xsd">
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
        <varString xsi:type="xsd:string">hello world</varString>
        <varStruct xsi:type="ns1:SOAPStruct">
          <varInt xsi:type="xsd:int">99</varInt>
          <varFloat xsi:type="xsd:float">4.0699e-12</varFloat>
          <varString xsi:type="xsd:string">nested struct</varString>
        </varStruct>
      </return>
    </test:echoNestedStructResponse>
  </env:Body>
</env:Envelope>
```

Test:T46

Description:

Node A sends a message to Node C in echoNestedArray containing an array with three elements. Node C responds with echoNestedArray.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoNestedArray xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputStruct xsi:type="ns1:SOAPArrayStruct"
                   xmlns:ns1="http://example.org/ts-tests/xsd">
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
        <varString xsi:type="xsd:string">hello world</varString>
        <varArray enc:itemType="xsd:string" enc:arraySize="3"
                          xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
          <item xsi:type="xsd:string">red</item>
          <item xsi:type="xsd:string">blue</item>
          <item xsi:type="xsd:string">green</item>
        </varArray>
      </inputStruct>
    </test:echoNestedArray>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoNestedArrayResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="ns1:SOAPArrayStruct">
              xmlns:ns1="http://example.org/ts-tests/xsd">
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
        <varString xsi:type="xsd:string">hello world</varString>
        <varArray enc:itemType="xsd:string" enc:arraySize="3"
                          xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
          <item xsi:type="xsd:string">red</item>
          <item xsi:type="xsd:string">blue</item>
          <item xsi:type="xsd:string">green</item>
        </varArray>
      </return>
    </test:echoNestedArrayResponse>
  </env:Body>
```

```
</env:Envelope>
```

Test:T47

Description:

Node A sends a message to Node C in echoFloatArray with enc:itemType="ns:float". Node C responds with echoFloatArray.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoFloatArray xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputFloatArray enc:itemType="xsd:float" enc:arraySize="2"
                       xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="xsd:float">0.00000555</item>
        <item xsi:type="xsd:float">12999.9</item>
      </inputFloatArray>
    </test:echoFloatArray>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoFloatArrayResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return enc:itemType="xsd:float" enc:arraySize="2"
                          xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="xsd:float">5.55E-06</item>
        <item xsi:type="xsd:float">12999.9</item>
      </return>
    </test:echoFloatArrayResponse>
  </env:Body>
</env:Envelope>
```

Test:T48

Description:

Node A sends a message to Node C in echoStringArray with enc:itemType="ns:string". Node C responds with echoStringArray.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStringArray xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputStringArray enc:itemType="xsd:string" enc:arraySize="2"
                        xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="xsd:string">hello</item>
        <item xsi:type="xsd:string">world</item>
      </inputStringArray>
    </test:echoStringArray>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStringArrayResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return enc:itemType="xsd:string" enc:arraySize="2"
              xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="xsd:string">hello</item>
        <item xsi:type="xsd:string">world</item>
      </return>
    </test:echoStringArrayResponse>
  </env:Body>
</env:Envelope>
```

Test:T49

Description:

Node A sends a message to Node C in echoStringArray without a enc:itemType attribute. Node C responds with echoStringArray.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStringArray xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputStringArray enc:arraySize="2"
                        xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="xsd:string">hello</item>
        <item xsi:type="xsd:string">world</item>
      </inputStringArray>
    </test:echoStringArray>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStringArrayResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return enc:itemType="xsd:string" enc:arraySize="2"
              xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="xsd:string">hello</item>
        <item xsi:type="xsd:string">world</item>
      </return>
    </test:echoStringArrayResponse>
  </env:Body>
</env:Envelope>
```

Test:T50

Description:

Node A sends a message to Node C in echoIntegerArray with enc:itemType="ns:integer".

Node C responds with echoIntegerArray.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoIntegerArray xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputIntegerArray enc:itemType="xsd:int" enc:arraySize="2"
                         xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="xsd:int">100</item>
        <item xsi:type="xsd:int">200</item>
      </inputIntegerArray>
    </test:echoIntegerArray>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoIntegerArrayResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return enc:itemType="xsd:int" enc:arraySize="2"
              xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="xsd:int">100</item>
        <item xsi:type="xsd:int">200</item>
      </return>
    </test:echoIntegerArrayResponse>
  </env:Body>
</env:Envelope>
```

Test:T51

Description:

Node A sends a message to Node C with Base64-encoded binary data. Node C responds with echoBase64.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
```

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <test:echoBase64 xmlns:test="http://example.org/ts-tests"
       env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputBase64 xsi:type="xsd:base64Binary">
        YUdWc2JHOGdkMjl5YkdRPQ==
      </inputBase64>
    </test:echoBase64>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <test:echoBase64Response xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:base64Binary">
        YUdWc2JHOGdkMjl5YkdRPQ==
      </return>
    </test:echoBase64Response>
  </env:Body>
</env:Envelope>
```

Test:T52

Description:

Node A sends a message to Node C containing a boolean datatype with value = 1 in echoBoolean. Note C responds with echoBoolean.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoBoolean xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputBoolean xsi:type="xsd:boolean">1</inputBoolean>
    </test:echoBoolean>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoBooleanResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:boolean">1</return>
    </test:echoBooleanResponse>
  </env:Body>
</env:Envelope>
```

Test:T53

Description:

Node A sends a message to Node C in echoDate with date datatype. Node C responds with echoDate.

Messages:

Message sent from Node A

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoDate xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputDate xsi:type="xsd:date">1956-10-18</inputDate>
    </test:echoDate>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoDateResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:date">1956-10-18</return>
    </test:echoDateResponse>
  </env:Body>
</env:Envelope>
```

Test:T54

Description:

Node A sends a message to Node C in echoDecimal with decimal datatype. Node C responds with echoDecimal.

Messages:

Message sent from Node A

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoDecimal xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputDecimal xsi:type="xsd:decimal">123.45678901234567890</inputDecimal>
    </test:echoDecimal>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoDecimalResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:decimal">123.4567890123456789</return>
    </test:echoDecimalResponse>
  </env:Body>
</env:Envelope>
```

Test:T55

Description:

Node A sends a message to Node C in echoFloat with floating point datatype. Node C responds with echoFloat.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoFloat xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputFloat xsi:type="xsd:float">0.005</inputFloat>
    </test:echoFloat>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoFloatResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:float">0.005</return>
    </test:echoFloatResponse>
  </env:Body>
</env:Envelope>
```

Test:T56

Description:

Node A sends to Node C a message with body containing a ref attribute information item referencing a non-existent id attribute information item. Node C responds with a fault.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
  <env:Header>
    <test:DataHolder xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <test:Data enc:id="data-1" xsi:type="xsd:string">
        hello world
      </test:Data>
    </test:DataHolder>
  </env:Header>
  <env:Body>
    <test:echoString xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString enc:ref="data-2" xsi:type="xsd:string" />
    </test:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
        <env:Subcode>
          <env:Value>enc:MissingID</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US"> Unresolved reference </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T57

Description:

Node A sends to Node C a message with body containing a ref attribute information item referencing a unique identifier defined by an id attribute information item. Node C responds by echoing the value of the element information item containing the referenced id attribute information item.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
  <env:Header>
    <test:DataHolder xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <test:Data enc:id="data" xsi:type="xsd:string">
        hello world
      </test:Data>
    </test:DataHolder>
  </env:Header>
  <env:Body>
    <test:echoString xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <test:inputString enc:ref="data" xsi:type="xsd:string" />
    </test:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <env:Body>
      <test:echoStringResponse xmlns:test="http://example.org/ts-tests"
            xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
            env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
        <rpc:result>return</rpc:result>
        <return xsi:type="xsd:string">
          hello world
        </return>
      </test:echoStringResponse>
    </env:Body>
  </env:Envelope>
```

Test:T58

Description:

Node A sends a message to Node C invoking the rpc echoIntegerArray. The inputIntegerArray element contains the attribute enc:itemType="xsd:int" but the children of this element are not of type xsd:int. Node C returns a fault.

Messages:

Message sent from Node A

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <env:Body>
      <test:echoIntegerArray xmlns:test="http://example.org/ts-tests"
            env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
        <inputIntegerArray enc:itemType="xsd:int" enc:arraySize="1"
                     xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
          <a><b>1</b></a>
        </inputIntegerArray>
      </test:echoIntegerArray>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
       <env:Fault>
```

```
        <env:Code>
          <env:Value>env:Sender</env:Value>
          <env:Subcode>
            <env:Value xmlns:rpc="http://www.w3.org/2003/05/soap-rpc">
              rpc:BadArguments
            </env:Value>
          </env:Subcode>
        </env:Code>
        <env:Reason>
         <env:Text xml:lang="en-US">
           Violation of encoding rules
         </env:Text>
        </env:Reason>
        <env:Detail>
           Array element declared as array of integers contains elements with
           wrong content.
       </env:Detail>
      </env:Fault>
   </env:Body>
 </env:Envelope>
```

Test:T59

Description:

Node A sends a message to Node C with a body containing a ref attribute information item and an id information item on the same element information item. Node C returns a fault.

Messages:

Message sent from Node A

```
 <?xml version="1.0"?>
 <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <env:Body>
     <test:echoStringArray xmlns:test="http://example.org/ts-tests"
           env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
       <inputStringArray enc:itemType="xsd:string"
                         xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
         <item enc:id="data" xsi:type="xsd:string" enc:ref="data">hello</item>
         <item>world</item>
       </inputStringArray>
     </test:echoStringArray>
   </env:Body>
 </env:Envelope>
```

Message sent from Node C

```
 <?xml version="1.0"?>
 <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
               xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
   <env:Body>
     <env:Fault>
       <env:Code>
         <env:Value>env:Sender</env:Value>
```

```
        <env:Subcode>
          <env:Value>enc:MissingID</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US">
          Violation of id and ref information items
        </env:Text>
      </env:Reason>
      <env:detail>
        A ref attribute information item and an id attribute information
        item MUST NOT appear on the same element information item.
      </env:detail>
    </env:Fault>
 </env:Body>
</env:Envelope>
```

Test:T60

Description:

Node A sends to Node C a message specifying an array with bound specified by an asterisk. Node C responds with the count of items that appeared in the input array.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:countItems xmlns:test="http://example.org/ts-tests"
          xmlns:enc="http://www.w3.org/2003/05/soap-encoding"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputStringArray enc:itemType="xsd:string" enc:arraySize="*">
        <item xsi:type="xsd:string">hello</item>
        <item xsi:type="xsd:string">world</item>
      </inputStringArray>
    </test:countItems>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:countItemsResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:int">
        2
      </return>
    </test:countItemsResponse>
  </env:Body>
</env:Envelope>
```

Test:T61

Description:

Node A sends to Node C a message containing an array with a list of dimensions consisting of an integer followed by an asterisk. Node C responds with a fault stating that the asterisk can occur only in the first position in the dimension list.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:countItems xmlns:test="http://example.org/ts-tests"
          xmlns:enc="http://www.w3.org/2003/05/soap-encoding"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputStringArray enc:itemType="xsd:string" enc:arraySize="2 *">
        <item xsi:type="xsd:string">hello</item>
        <item xsi:type="xsd:string">world</item>
      </inputStringArray>
    </test:countItems>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US">
          * may only be first arraySize value in list
        </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T62

Description:

Node A sends a msg to node B with three headers in it with MU=1 and targeted at Node B. The semantics of processing the header test:concatAndForwardEchoOK requires Node B to take the contents of test:concatAndForwardEchoOkArg1 and test:concatAndForwardEchoOkArg2, concatenate them and forward the result to Node C using the test:echoOk header. Node C then receives the concetanated data in the test:echoOk header and responds using the test:responsOk header.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:concatAndForwardEchoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://example.org/ts-tests/B"
          env:mustUnderstand="1"/>
    <test:concatAndForwardEchoOkArg1 xmlns:test="http://example.org/ts-tests"
          env:role="http://example.org/ts-tests/B"
          env:mustUnderstand="1">
      StringA
    </test:concatAndForwardEchoOkArg1>
    <test:concatAndForwardEchoOkArg2 xmlns:test="http://example.org/ts-tests"
          env:role="http://example.org/ts-tests/B"
          env:mustUnderstand="1">
      StringB
    </test:concatAndForwardEchoOkArg2>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node B

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://example.org/ts-tests/C"
          env:mustUnderstand="1">
      StringAStringB
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
```

```
    <env:Header>
      <test:responseOk xmlns:test="http://example.org/ts-tests">
        StringAStringB
      </test:responseOk>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Test:T63

Description:

Node A sends a message to node C with one headers in it with MU=1 and targeted at Node C. The semantics of processing the header test:validateCountryCode requires Node C to take the contents of test:validateCountryCode and validate that it consists of 2 letters. If the code sent is not 2 letters then a fault is sent back. The details of the fault are sent in the header test:validateCountryCodeFault

Messages:

Message sent from Node A

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:validateCountryCode xmlns:test="http://example.org/ts-tests"
            env:role="http://example.org/ts-tests/C"
            env:mustUnderstand="1">
        ABCD
      </test:validateCountryCode>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version='1.0' ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <test:validateCountryCodeFault xmlns:test="http://example.org/ts-tests">
        Country code must be 2 letters.
      </test:validateCountryCodeFault>
    </env:Header>
    <env:Body>
       <env:Fault>
         <env:Code>env:Sender</env:Code>
         <env:Reason>
           <env:Text xml:lang="en-US">
             Not a valid country code
           </env:Text>
         </env:Reason>
       </env:Fault>
    </env:Body>
  </env:Envelope>
```

Test:T64

Description:

Node A sends a message to node C with non-empty [notation] property. Node C responds by sending back a fault.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<!DOCTYPE foo [
<!NOTATION application_xml SYSTEM 'http://www.isi.edu/in-notes/iana/assignments/me
]>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Body>
    <test:echoOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:echoOk>
 </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
     <env:Fault>
       <env:Code>
        <env:Value>env:Sender</env:Value>
       </env:Code>
       <env:Reason>
        <env:Text xml:lang="en-US">
          DTD are not supported by SOAP 1.2
        </env:Text>
       </env:Reason>
     </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T65

Description:

Node A sends a message to node C with non-empty [unparsed entity] property. Node C responds by sending back a fault.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<!DOCTYPE foo [
<!ELEMENT Envelope (Body) >
<!ELEMENT Body (echoOk) >
<!ELEMENT echoOk (#PCDATA) >
]>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <test:echoOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:echoOk>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
     <env:Fault>
       <env:Code>
        <env:Value>env:Sender</env:Value>
       </env:Code>
       <env:Reason>
        <env:Text xml:lang="en-US">
          DTD are not supported by SOAP 1.2
        </env:Text>
       </env:Reason>
     </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T66

Description:

Node A sends a message to node C with non-empty [encoding] property. Node C responds by sending the appropriate 'responseOk'.

Messages:

Message sent from Node A

```
<?xml version='1.0' encoding='UTF-8'?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:responseOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:responseOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Test:T67

Description:

Node A sends a message to node C with [standalone] property as 'true'. Node C responds by sending the appropriate 'responseOk'.

Messages:

Message sent from Node A

```
<?xml version='1.0' standalone='yes'?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <test:echoOk xmlns:test="http://example.org/ts-tests"
        env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
    foo
  </test:echoOk>
 </env:Header>
 <env:Body>
 </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:responseOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:responseOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Test:T68

Description:

Node A sends a message to node C with a soap message which is semantically eqivalent to the request message in T1 test (the only difference being additional whitespace character information item). Node C responds by sending the appropriate 'responseOk'.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
```

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">


 <env:Header           >

                           <test:echoOk xmlns:test="http://example.org/ts-tests"
        env:role="http://www.w3.org/2003/05/soap-envelope/role/next"  >
    foo
  </test:echoOk>


 </env:Header>
 <env:Body>


 </env:Body>




</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:responseOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:responseOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Test:T69

Description:

Node A sends a message to node C which does not contain a Body element. Node C responds by sending an appropriate fault

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
        <test:echoOk xmlns:test="http://example.org/ts-tests">foo</test:echoOk>
 </env:Header>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
```

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
     <env:Fault>
       <env:Code>
        <env:Value>env:Sender</env:Value>
       </env:Code>
       <env:Reason>
        <env:Text xml:lang="en-US">
          env:Body must be present in a SOAP 1.2 envelope
        </env:Text>
       </env:Reason>
     </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T70

Description:

Node A sends a message to node C which contains Header, Body and a Trailer element
Node C responds by sending an appropriate fault

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
        <test:echoOk xmlns:test="http://example.org/ts-tests">foo</test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
  <Trailer>
  </Trailer>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
     <env:Fault>
       <env:Code>
        <env:Value>env:Sender</env:Value>
       </env:Code>
       <env:Reason>
        <env:Text xml:lang="en-US">
          A SOAP 1.2 envelope can contain only Header and Body
        </env:Text>
       </env:Reason>
     </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T71

Description:

Node A sends a message to node C with a non-namespace qualified attribute on the Envelope Node C responds by sending an appropriate fault

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
     attr1="a-value">
  <env:Header>
        <test:echoOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:echoOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
     <env:Fault>
       <env:Code>
        <env:Value>env:Sender</env:Value>
       </env:Code>
       <env:Reason>
        <env:Text xml:lang="en-US">
          A SOAP 1.2 Envelope element cannot have non Namespace qualified
          attributes
        </env:Text>
       </env:Reason>
     </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T72

Description:

Node A sends a message to node C with the encodingStyle set on the Envelope element. Node C responds by sending an appropriate fault

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  <env:Body>
    <test:echoOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:echoOk>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US">
          env:encodingStyle cannot be specified on the env:Envelope
        </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:T73

Description:

Node A sends a message to node C, using the SOAP RPC convention. The RCP asks for node C to echo back the string. The message sent by A has encodingStyle set on the Body block as well as the RPC input parameter. Node C responds by echoing the parameter.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoString xmlns:test="http://example.org/ts-tests"
                     env:encodingStyle="http://www.w3.org/2003/05/soap-envelope/en
      <test:inputString xsi:type="xsd:string"
            env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
        hello world
      </test:inputString>
    </test:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStringResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:string">hello world
      </return>
    </test:echoStringResponse>
  </env:Body>
</env:Envelope>
```

Test:T74

Description:

Node A sends a message to Node C with 2 headers echoOk and Unknown. The Unknown header does not have a mustUnderstand or role attribute set, but the child of the header block does. Node C does not understand the Unknown header but can safely ignore it, and responds to the echoOk header.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
      foo
    </test:echoOk>
    <test:Unknown xmlns:test="http://example.org/ts-tests">
      <test:raiseFault env:mustUnderstand="1"
            env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
      </test:raiseFault>
    </test:Unknown>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:responseOk xmlns:test="http://example.org/ts-tests">
      foo
    </test:responseOk>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Test:T75

Description:

Node A sends a message to Node C with header echoResolvedRef containing a relative reference defined by xlink:href and xml:base. Node C responds by echoing the resolved reference.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <test:echoResolvedRef xmlns:test="http://example.org/ts-tests"
        env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
        env:mustUnderstand="1">
    <test:RelativeReference xml:base="http://example.org/today/"
          xlink:href="new.xml"
          xmlns:xlink="http://www.w3.org/1999/xlink" />
  </test:echoResolvedRef>
 </env:Header>
 <env:Body>
 </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
```

```
    <env:Header>
      <test:responseResolvedRef xmlns:test="http://example.org/ts-tests">
        http://example.org/today/new.xml
      </test:responseResolvedRef>
    </env:Header>
    <env:Body>
    </env:Body>
  </env:Envelope>
```

Test:T76

Description:

Node A sends 2 different serialization for echoString test and gets the same response back from Node C

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoString xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">
        hello world
      </inputString>
    </test:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStringResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:string">
        hello world
      </return>
    </test:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node A

```
<?xml version='1.0' ?>
```

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
  <env:Header>
    <test:DataHolder xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <test:Data enc:id="data" xsi:type="xsd:string">
        hello world
      </test:Data>
    </test:DataHolder>
  </env:Header>
  <env:Body>
    <test:echoString xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString enc:ref="data" xsi:type="xsd:string" />
    </test:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStringResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:string">
        hello world
      </return>
    </test:echoString>
  </env:Body>
</env:Envelope>
```

Test:T77

Description:

This test consists of an RPC called 'isNil'. This RPC has one parameter of any type. The return value is of type boolean. If the request message has a parameter that was absent or had xsi:nil='1', the returned value is true, else it is false. Node A is the requesting node and Node C is the responding node. This test consists of 3 requests and responses send by Node A and Node C respectively.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
  <env:Body>
    <test:isNil xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:nil="1" />
    </test:isNil>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:isNilResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:boolean">
        1
      </return>
    </test:isNilResponse>
  </env:Body>
</env:Envelope>
```

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:isNil xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
    </test:isNil>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:isNilResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:boolean">
        1
      </return>
    </test:isNilResponse>
  </env:Body>
</env:Envelope>
```

Message sent from Node A

```
<?xml version='1.0' ?>
```

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:isNil xmlns:test="http://example.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">
        This is a string
      </inputString>
    </test:isNil>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:isNilResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:boolean">
        0
      </return>
    </test:isNilResponse>
  </env:Body>
</env:Envelope>
```

Test:T78

Description:

Node A sends to node C message with echoOk header block having role equal to "http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver". NodeC returns back empty body with responseOK header.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <test:echoOk xmlns:test="http://example.org/ts-tests"
        env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver">
    foo
  </test:echoOk>
 </env:Header>
 <env:Body>
 </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <test:responseOk xmlns:test="http://example.org/ts-tests">
    foo
  </test:responseOk>
 </env:Header>
 <env:Body>
 </env:Body>
</env:Envelope>
```

Test:T79

Description:

Node A sends to node B message with echoOk header block having role equal to "http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver". Node B forwards this message to Node C. NodeC returns back empty body with responseOK header.

Messages:

Message sent from Node A

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <test:echoOk xmlns:test="http://example.org/ts-tests"
        env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver">
    foo
  </test:echoOk>
 </env:Header>
 <env:Body>
 </env:Body>
</env:Envelope>
```

Message sent from Node B

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <test:echoOk xmlns:test="http://example.org/ts-tests"
        env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver">
    foo
  </test:echoOk>
 </env:Header>
 <env:Body>
 </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <test:responseOk xmlns:test="http://example.org/ts-tests">
    foo
  </test:responseOk>
 </env:Header>
 <env:Body>
 </env:Body>
</env:Envelope>
```

Test:T80

Description:

This tests assumes the existence of a conventionally non-existent encoding to test a failure condition. It is the intention that no receiving SOAP 1.2 node will support this encoding and that on receiving this encoding style declaration in a SOAP message, will return the DataEncodingUnknown fault. Node A sends a simple SOAP 1.2 message to Node C containing the poison encoding style declaration. Node C responds with a fault.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <test:echoOk env:encodingStyle="http://example.org/PoisonEncoding"
                 xmlns:test="http://example.org/ts-tests">
      foo
    </test:echoOk>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:DataEncodingUnknown</env:value>
       </env:Code>
       <env:Reason>
        <env:Text xml:lang="en-US"> Unknown Data Encoding Style </env:Text>
       </env:Reason>
     </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:TH1

Description:

This tests consists of testing SOAP 1.2 HTTP bindings. Node A sends a valid SOAP request message over HTTP 1.1 to Node C. Node C responds with a HTTP status 200 and a SOAP response message.

Messages:

Message sent from Node A

```
POST /soap1.2/interop HTTP/1.1
Host: www.w3.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoString xmlns:test="http://example.org/ts-tests">
      <inputString xsi:type="xsd:string">hello world</inputString>
    </test:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoStringResponse xmlns:test="http://example.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:string">hello world</return>
    </test:echoStringResponse>
  </env:Body>
</env:Envelope>
```

Test:TH2

Description:

This tests consists of testing SOAP 1.2 HTTP bindings. Node A sends an invalid SOAP (with a missing 'Body' element) request message over HTTP 1.1 to Node C. Node C responds with a HTTP status 400 and a SOAP message containing a 'env:Sender' SOAP fault.

Messages:

Message sent from Node A

```
POST /soap1.2/interop HTTP/1.1
Host: www.w3.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:echoOk xmlns:test="http://example.org/ts-tests">foo</test:echoOk>
  </env:Header>
</env:Envelope>
```

Message sent from Node C

```
HTTP/1.1 400 Bad Request
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US">
          env:Body must be present in a SOAP 1.2 envelope
        </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:TH3

Description:

This tests consists of testing SOAP 1.2 HTTP bindings. Node A sends a valid SOAP 1.1 request message over HTTP 1.1 to Node C. Node C responds with a HTTP status 500 and a SOAP message containing a 'env:VersionMismatch' SOAP fault.

Messages:

Message sent from Node A

```
POST /soap1.2/interop HTTP/1.1
Host: www.w3.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope">
  <env:Body>
    <test:echoOk xmlns:test="http://example.org/ts-tests">foo</test:echoOk>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <env:Upgrade>
      <env:SupportedEnvelope qname="ns1:Envelope"
                    xmlns:ns1="http://www.w3.org/2003/05/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-US">Wrong Version</env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:TH4

Description:

This tests consists of testing SOAP 1.2 HTTP bindings. Node A sends a SOAP request message over HTTP 1.1 to Node C. This request message contains a header with mustUnderstand="1", which Node C does not understand. Node C responds with a HTTP status 500 and a SOAP message containing a 'env:MustUnderstand' SOAP fault.

Messages:

Message sent from Node A

```
POST /soap1.2/interop HTTP/1.1
Host: www.w3.org
Content-Type: application/soap+xml; charset="utf-8"
```

```
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <test:Unknown xmlns:test="http://example.org/ts-tests"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
          env:mustUnderstand="1">
      foo
    </test:Unknown>
  </env:Header>
  <env:Body>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <env:NotUnderstood qname="test:Unknown" xmlns:test="http://example.org/ts-test
  </env:Header>
  <env:Body>
     <env:Fault>
       <env:Code>
        <env:Value>env:MustUnderstand</env:Value>
       </env:Code>
       <env:Reason>
        <env:Text xml:lang="en-US">
          Header not understood
        </env:Text>
       </env:Reason>
       <env:Role>http://www.w3.org/2003/05/soap-envelope/role/next</env:Role>
     </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:TH5

Description:

This tests consists of testing SOAP 1.2 HTTP bindings. Node A sends an valid SOAP request message over HTTP 1.1 to Node C, using media type "audio/mpeg". Node C responds with a HTTP status 415.

Messages:

Message sent from Node A

```
POST /soap1.2/interop HTTP/1.1
Host: www.w3.org
Content-Type: audio/mpeg
```

```
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoString xmlns:test="http://example.org/ts-tests">
      <inputString xsi:type="xsd:string">hello world</inputString>
    </test:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
HTTP/1.1 415 Unsupported Media
```

Test:SBR1-echoString

Description:

This test is the 'echoString' test defined by SOAPBuilders Round 1 [SOAPBuilders Round 1] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">Hello world</inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoStringResponse xmlns:sb="http://soapinterop.org/"
        xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:string">Hello world</return>
    </sb:echoStringResponse>
  </env:Body>
</env:Envelope>
```

Test:SBR1-echoStringArray

Description:

This test is the 'echoStringArray' test defined by SOAPBuilders Round 1 [SOAPBuilders Round 1] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoStringArray xmlns:sb="http://soapinterop.org/"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputStringArray enc:itemType="xsd:string" enc:arraySize="2"
                        xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="xsd:string">hello</item>
        <item xsi:type="xsd:string">world</item>
      </inputStringArray>
    </sb:echoStringArray>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoStringArrayResponse xmlns:sb="http://soapinterop.org/"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return enc:itemType="xsd:string" enc:arraySize="2"
              xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="xsd:string">hello</item>
        <item xsi:type="xsd:string">world</item>
      </return>
    </sb:echoStringArrayResponse>
  </env:Body>
</env:Envelope>
```

Test:SBR1-echoInteger

Description:

This test is the 'echoInteger' test defined by SOAPBuilders Round 1 [SOAPBuilders Round 1] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoInteger xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputInteger xsi:type="xsd:int">123</inputInteger>
    </sb:echoInteger>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoIntegerResponse xmlns:sb="http://soapinterop.org/"
        xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:int">123</return>
    </sb:echoIntegerResponse>
  </env:Body>
</env:Envelope>
```

Test:SBR1-echoIntegerArray

Description:

This test is the 'echoIntegerArray' test defined by SOAPBuilders Round 1 [SOAPBuilders Round 1] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoIntegerArray xmlns:sb="http://soapinterop.org/"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputIntegerArray enc:itemType="xsd:int" enc:arraySize="2"
                         xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="xsd:int">100</item>
        <item xsi:type="xsd:int">200</item>
      </inputIntegerArray>
    </sb:echoIntegerArray>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoIntegerArrayResponse xmlns:sb="http://soapinterop.org/"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return enc:itemType="xsd:int" enc:arraySize="2"
              xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="xsd:int">100</item>
        <item xsi:type="xsd:int">200</item>
      </return>
    </sb:echoIntegerArrayResponse>
  </env:Body>
</env:Envelope>
```

Test:SBR1-echoFloat

Description:

This test is the 'echoFloat' test defined by SOAPBuilders Round 1 [SOAPBuilders Round 1] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoFloat xmlns:sb="http://soapinterop.org/"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputFloat xsi:type="xsd:float">0.005</inputFloat>
    </sb:echoFloat>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoFloatResponse xmlns:sb="http://soapinterop.org/"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:float">0.005</return>
    </sb:echoFloatResponse>
  </env:Body>
</env:Envelope>
```

Test:SBR1-echoFloatArray

Description:

This test is the 'echoFloatArray' test defined by SOAPBuilders Round 1 [SOAPBuilders Round 1] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoFloatArray xmlns:sb="http://soapinterop.org/"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputFloatArray enc:itemType="xsd:float" enc:arraySize="2"
                       xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="xsd:float">0.00000555</item>
        <item xsi:type="xsd:float">12999.9</item>
      </inputFloatArray>
    </sb:echoFloatArray>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoFloatArrayResponse xmlns:sb="http://soapinterop.org/"
```

```
            xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
            env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
        <rpc:result>return</rpc:result>
        <return enc:itemType="xsd:float" enc:arraySize="2"
                                xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
          <item xsi:type="xsd:float">5.55E-06</item>
          <item xsi:type="xsd:float">12999.9</item>
        </return>
      </sb:echoFloatArrayResponse>
    </env:Body>
  </env:Envelope>
```

Test:SBR1-echoStruct

Description:

This test is the 'echoStruct' test defined by SOAPBuilders Round 1 [SOAPBuilders Round 1] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
  <?xml version="1.0"?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <env:Body>
      <sb:echoStruct xmlns:sb="http://soapinterop.org/"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
        <inputStruct xsi:type="ns1:SOAPStruct"
                                              xmlns:ns1="http://soapinterop.org/xsd">
          <varInt xsi:type="xsd:int">42</varInt>
          <varFloat xsi:type="xsd:float">0.005</varFloat>
          <varString xsi:type="xsd:string">hello world</varString>
        </inputStruct>
      </sb:echoStruct>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version="1.0"?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <env:Body>
                <sb:echoStructResponse xmlns:sb="http://soapinterop.org/"
            xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
            env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
        <rpc:result>return</rpc:result>
        <return xsi:type="ns1:SOAPStruct"
                                         xmlns:ns1="http://soapinterop.org/xsd">
          <varInt xsi:type="xsd:int">42</varInt>
          <varFloat xsi:type="xsd:float">0.005</varFloat>
          <varString xsi:type="xsd:string">hello world</varString>
        </return>
      </sb:echoStructResponse>
    </env:Body>
  </env:Envelope>
```

Test:SBR1-echoStructArray

Description:

This test is the 'echoStructArray' test defined by SOAPBuilders Round 1 [SOAPBuilders Round 1] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoStructArray xmlns:sb="http://soapinterop.org/"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputStructArray enc:itemType="ns1:SOAPStruct"
                        enc:arraySize="2"
                        xmlns:ns1="http://soapinterop.org/xsd"
                        xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="ns1:SOAPStruct">
          <varInt xsi:type="xsd:int">42</varInt>
          <varFloat xsi:type="xsd:float">0.005</varFloat>
          <varString xsi:type="xsd:string">hello world</varString>
        </item>
        <item xsi:type="ns1:SOAPStruct">
          <varInt xsi:type="xsd:int">43</varInt>
          <varFloat xsi:type="xsd:float">0.123</varFloat>
          <varString xsi:type="xsd:string">bye world</varString>
        </item>
      </inputStructArray>
    </sb:echoStructArray>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoStructArrayResponse xmlns:sb="http://soapinterop.org/"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return enc:itemType="ns1:SOAPStruct"
              enc:arraySize="2"
              xmlns:ns1="http://soapinterop.org/xsd"
              xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item xsi:type="ns1:SOAPStruct">
```

```
          <varInt xsi:type="xsd:int">42</varInt>
          <varFloat xsi:type="xsd:float">0.005</varFloat>
          <varString xsi:type="xsd:string">hello world</varString>
        </item>
        <item xsi:type="ns1:SOAPStruct">
          <varInt xsi:type="xsd:int">43</varInt>
          <varFloat xsi:type="xsd:float">0.123</varFloat>
          <varString xsi:type="xsd:string">bye world</varString>
        </item>
      </return>
    </sb:echoStructArrayResponse>
  </env:Body>
</env:Envelope>
```

Test:SBR1-echoVoid

Description:

This test is the 'echoVoid' test defined by SOAPBuilders Round 1 [SOAPBuilders Round 1] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoVoid xmlns:sb="http://soapinterop.org/"
              env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"/>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoVoidResponse xmlns:sb="http://soapinterop.org/"
              env:encodingStyle="http://www.w3.org/2003/05/soap-encoding" />
  </env:Body>
</env:Envelope>
```

Test:SBR1-echoBase64

Description:

This test is the 'echoBase64' test defined by SOAPBuilders Round 1 [SOAPBuilders Round 1] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <sb:echoBase64 xmlns:test="http://soapinterop.org/"
       env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputBase64 xsi:type="xsd:base64Binary">
        YUdWc2JHOGdkMjl5YkdRPQ==
      </inputBase64>
    </sb:echoBase64>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <sb:echoBase64Response xmlns:sb="http://soapinterop.org/"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:base64Binary">
        YUdWc2JHOGdkMjl5YkdRPQ==
      </return>
    </sb:echoBase64Response>
  </env:Body>
</env:Envelope>
```

Test:SBR1-echoDate

Description:

This test is the 'echoDate' test defined by SOAPBuilders Round 1 [SOAPBuilders Round 1] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoDate xmlns:test="http://soapinterop.org/"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputDate xsi:type="xsd:dateTime">1956-10-18T22:20:00-07:00</inputDate>
    </sb:echoDate>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoDateResponse xmlns:sb="http://soapinterop.org/"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:dateTime">1956-10-19T05:20:00Z</return>
    </sb:echoDateResponse>
  </env:Body>
</env:Envelope>
```

Test:SBR2-echoHexBinary

Description:

This test is the 'echoHexBinary' test defined by SOAPBuilders Round 2 [SOAPBuilders Round 2] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <sb:echoHexBinary xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputHexBinary xsi:type="xsd:hexBinary">
        68656C6C6F20776F726C6421
      </inputHexBinary>
    </sb:echoHexBinary>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <sb:echoHexBinaryResponse xmlns:sb="http://soapinterop.org/"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:hexBinary">
        68656C6C6F20776F726C6421
      </return>
    </sb:echoHexBinaryResponse>
  </env:Body>
</env:Envelope>
```

Test:SBR2-echoDecimal

Description:

This test is the 'echoDecimal' test defined by SOAPBuilders Round 2 [SOAPBuilders Round 2] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <sb:echoDecimal xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputDecimal xsi:type="xsd:decimal">
        123.45678901234567890
      </inputDecimal>
    </sb:echoDecimal>
  </env:Body>
```

```
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <sb:echoDecimalResponse xmlns:sb="http://soapinterop.org/"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:decimal">
        123.45678901234567890
      </return>
    </sb:echoDecimalResponse>
  </env:Body>
</env:Envelope>
```

Test:SBR2-echoBoolean

Description:

This test is the 'echoBoolean' test defined by SOAPBuilders Round 2 [SOAPBuilders Round 2] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <sb:echoBoolean xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
        <inputBoolean xsi:type="xsd:boolean">1</inputBoolean>
    </sb:echoBoolean>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <sb:echoBooleanResponse xmlns:sb="http://soapinterop.org/"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:boolean">1</return>
    </sb:echoBooleanResponse>
  </env:Body>
</env:Envelope>
```

Test:SBR2-echoStructAsSimpleTypes

Description:

This test is the 'echoStructAsSimpleTypes' test defined by SOAPBuilders Round 2 Group B [SOAPBuilders Round 2 Group B] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
  <?xml version="1.0"?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <env:Body>
      <sb:echoStructAsSimpleTypes xmlns:sb="http://soapinterop.org/"
            env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
        <inputStruct xsi:type="ns1:SOAPStruct"
                     xmlns:ns1="http://soapinterop.org/xsd">
          <varString xsi:type="xsd:string">hello world</varString>
          <varInt xsi:type="xsd:int">42</varInt>
          <varFloat xsi:type="xsd:float">0.005</varFloat>
        </inputStruct>
      </sb:echoStructAsSimpleTypes>
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version="1.0"?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <env:Body>
      <sb:echoStructAsSimpleTypesResponse xmlns:sb="http://soapinterop.org/"
            env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
        <outputString xsi:type="xsd:string">hello world</outputString>
        <outputInt xsi:type="xsd:int">42</outputInt>
        <outputFloat xsi:type="xsd:float">0.005</outputFloat>
      </sb:echoStructAsSimpleTypesResponse>
    </env:Body>
  </env:Envelope>
```

Test:SBR2-echoSimpleTypesAsStruct

Description:

This test is the 'echoSimpleTypesAsStruct' test defined by SOAPBuilders Round 2 Group B [SOAPBuilders Round 2 Group B] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoSimpleTypesAsStruct xmlns:sb="http://soapinterop.org/"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">hello world</inputString>
      <inputInt xsi:type="xsd:int">42</inputInt>
      <inputFloat xsi:type="xsd:float">0.005</inputFloat>
    </sb:echoSimpleTypesAsStruct>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoSimpleTypesAsStructResponse xmlns:sb="http://soapinterop.org/"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="ns1:SOAPStruct"
              xmlns:ns1="http://soapinterop.org/xsd">
        <varString xsi:type="xsd:string">hello world</varString>
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
      </return>
    </sb:echoSimpleTypesAsStructResponse>
  </env:Body>
</env:Envelope>
```

Test:SBR2-echo2DStringArray

Description:

This test is the 'echo2DStringArray' test defined by SOAPBuilders Round 2 Group B [SOAPBuilders Round 2 Group B] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echo2DStringArray xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <input2DStringArray enc:itemType="xsd:string"
                          enc:arraySize="2 3"
                          xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item>row0col0</item>
        <item>row0col1</item>
        <item>row0col2</item>
        <item>row1col0</item>
        <item>row1col1</item>
        <item>row1col2</item>
      </input2DStringArray>
    </sb:echo2DStringArray>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echo2DStringArrayResponse xmlns:sb="http://soapinterop.org/"
        xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <result enc:itemType="xsd:string"
              enc:arraySize="2 3"
              xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
        <item>row0col0</item>
        <item>row0col1</item>
        <item>row0col2</item>
        <item>row1col0</item>
        <item>row1col1</item>
        <item>row1col2</item>
      </result>
    </sb:echo2DStringArrayResponse>
  </env:Body>
</env:Envelope>
```

Test:SBR2-echoNestedStruct

Description:

This test is the 'echoNestedStruct' test defined by SOAPBuilders Round 2 Group B [SOAPBuilders Round 2 Group B] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoNestedStruct xmlns:sb="http://soapinterop.org/"
       env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputStruct xsi:type="ns1:SOAPStructStruct"
        xmlns:ns1="http://soapinterop.org/xsd">
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
        <varString xsi:type="xsd:string">hello world</varString>
        <varStruct xsi:type="ns1:SOAPStruct">
          <varInt xsi:type="xsd:int">99</varInt>
          <varFloat xsi:type="xsd:float">4.0699e-12</varFloat>
          <varString xsi:type="xsd:string">nested struct</varString>
        </varStruct>
      </inputStruct>
    </sb:echoNestedStruct>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoNestedStructResponse xmlns:sb="http://soapinterop.org/"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="ns1:SOAPStructStruct"
              xmlns:ns1="http://soapinterop.org/xsd">
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
        <varString xsi:type="xsd:string">hello world</varString>
        <varStruct xsi:type="ns1:SOAPStruct">
          <varInt xsi:type="xsd:int">99</varInt>
          <varFloat xsi:type="xsd:float">4.0699e-12</varFloat>
          <varString xsi:type="xsd:string">nested struct</varString>
        </varStruct>
      </return>
    </sb:echoNestedStructResponse>
  </env:Body>
</env:Envelope>
```

Test:SBR2-echoNestedArray

Description:

This test is the 'echoNestedArray' test defined by SOAPBuilders Round 2 Group B [SOAPBuilders Round 2 Group B] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoNestedArray xmlns:sb="http://soapinterop.org/"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputStruct xsi:type="ns1:SOAPArrayStruct"
                   xmlns:ns1="http://soapinterop.org/xsd">
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
        <varString xsi:type="xsd:string">hello world</varString>
        <varArray enc:itemType="xsd:string" enc:arraySize="3"
                         xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
          <item xsi:type="xsd:string">red</item>
          <item xsi:type="xsd:string">blue</item>
          <item xsi:type="xsd:string">green</item>
        </varArray>
      </inputStruct>
    </sb:echoNestedArray>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoNestedArrayResponse xmlns:sb="http://soapinterop.org/"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="ns1:SOAPArrayStruct">
              xmlns:ns1="http://soapinterop.org/xsd">
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
        <varString xsi:type="xsd:string">hello world</varString>
        <varArray enc:itemType="xsd:string" enc:arraySize="3"
                         xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
          <item xsi:type="xsd:string">red</item>
          <item xsi:type="xsd:string">blue</item>
          <item xsi:type="xsd:string">green</item>
        </varArray>
      </return>
    </sb:echoNestedArrayResponse>
  </env:Body>
```

```
</env:Envelope>
```

Test:SBR2-echoMeStringRequest

Description:

This test is the 'echoMeStringRequest' test defined by SOAPBuilders Round 2 Group C [SOAPBuilders Round 2 Group C] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <h:echoMeStringRequest xmlns:h="http://soapinterop.org/echoheader/"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
         hello world
    </h:echoMeStringRequest>
  </env:Header>
  <env:Body>
    <sb:echoVoid xmlns:sb="http://soapinterop.org/"
        env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <h:echoMeStringResponse xmlns:h="http://soapinterop.org/echoheader/">
      hello world
    </h:echoMeStringResponse>
  </env:Header>
  <env:Body>
    <sb:echoVoidResponse xmlns:sb="http://soapinterop.org/"
        env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
  </env:Body>
</env:Envelope>
```

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <h:echoMeStringRequest xmlns:h="http://soapinterop.org/echoheader/"
       env:role="http://example.org/some/other/role">
         hello world
    </h:echoMeStringRequest>
  </env:Header>
  <env:Body>
    <sb:echoVoid xmlns:sb="http://soapinterop.org/"
        env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
  <?xml version="1.0"?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
      <sb:echoVoidResponse xmlns:sb="http://soapinterop.org/"
          env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
    </env:Body>
  </env:Envelope>
```

Test:SBR2-echoMeStructRequest

Description:

This test is the 'echoMeStructRequest' test defined by SOAPBuilders Round 2 Group C [SOAPBuilders Round 2 Group C] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
  <?xml version="1.0" ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <h:echoMeStructRequest xmlns:h="http://soapinterop.org/echoheader/"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="1">
        <varInt>42</varInt>
        <varFloat>99.005</varFloat>
        <varString>hello world</varString>
      </h:echoMeStructRequest>
    </env:Header>
    <env:Body>
      <sb:echoVoid xmlns:sb="http://soapinterop.org/"
          env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version="1.0"?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <h:echoMeStructResponse xmlns:h="http://soapinterop.org/echoheader/">
        <varInt>42</varInt>
        <varFloat>99.005</varFloat>
        <varString>hello world</varString>
      </h:echoMeStructResponse>
    </env:Header>
    <env:Body>
      <sb:echoVoidResponse xmlns:sb="http://soapinterop.org/"
          env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
    </env:Body>
  </env:Envelope>
```

Message sent from Node A

```
  <?xml version="1.0" ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <h:echoMeStructRequest xmlns:h="http://soapinterop.org/echoheader/"
         env:role="http://example.org/some/other/role"
         env:mustUnderstand="1">
        <varInt>42</varInt>
        <varFloat>99.005</varFloat>
        <varString>hello world</varString>
      </h:echoMeStructRequest>
    </env:Header>
    <env:Body>
      <sb:echoVoid xmlns:sb="http://soapinterop.org/"
          env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version="1.0"?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
      <sb:echoVoidResponse xmlns:sb="http://soapinterop.org/"
          env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
    </env:Body>
  </env:Envelope>
```

Test:SBR2-echoMeUnknown

Description:

This test is the 'echoMeUnknown' test defined by SOAPBuilders Round 2 Group C [SOAPBuilders Round 2 Group C] ported to SOAP 1.2.

Messages:

Message sent from Node A

```
  <?xml version="1.0" ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <h:echoMeUnknown xmlns:h="http://unknown.example.org/unknown/"
           env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
           nobody understands me!
      </h:echoMeUnknown>
    </env:Header>
    <env:Body>
      <sb:echoVoid xmlns:sb="http://soapinterop.org/"
          env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
    </env:Body>
```

```
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <sb:echoVoidResponse xmlns:sb="http://soapinterop.org/"
        env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
  </env:Body>
</env:Envelope>
```

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <h:echoMeUnknown xmlns:h="http://unknown.example.org/unknown/"
         env:mustUnderstand="1"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
         nobody understands me!
    </h:echoMeUnknown>
  </env:Header>
  <env:Body>
    <sb:echoVoid xmlns:sb="http://soapinterop.org/"
        env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <env:NotUnderstood qname="h:echoMeUnknown"
         xmlns:h="http://unknown.example.org/unknown/"/>
  </env:Header>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:MustUnderstand</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US"> Header not understood </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <h:echoMeUnknown xmlns:h="http://unknown.example.org/unknown/"
        env:role="http://example.org/some/other/role">
         nobody understands me!
    </h:echoMeUnknown>
  </env:Header>
  <env:Body>
```

```
      <sb:echoVoid xmlns:sb="http://soapinterop.org/"
          env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version="1.0"?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
      <sb:echoVoidResponse xmlns:sb="http://soapinterop.org/"
          env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
    </env:Body>
  </env:Envelope>
```

Message sent from Node A

```
  <?xml version="1.0" ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
      <h:echoMeUnknown xmlns:h="http://unknown.example.org/unknown/"
          env:mustUnderstand="1"
          env:role="http://example.org/some/other/role">
           nobody understands me!
      </h:echoMeUnknown>
    </env:Header>
    <env:Body>
      <sb:echoVoid xmlns:sb="http://soapinterop.org/"
          env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
  <?xml version="1.0"?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
      <sb:echoVoidResponse xmlns:sb="http://soapinterop.org/"
          env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
    </env:Body>
  </env:Envelope>
```

Test:XMLP-1

Description:

The test is based on SOAPBuilders Round 2 echoString test. But instead of the node A sending

<m:echoString xmlns:m="http://soapinterop.org/">
<inputString>hello world</inputString>
</m:echoString>

it sends

<m:echoString xmlns:m="http://soapinterop.org/"/>

The result is either (i) a response with the the missing input parameter missing in the output as well or with the output parameter having some default value determined by the server, or (ii) a SOAP fault with a Value of "env:Sender" for Code and a Value of "rpc:BadArguments" for Subcode.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"/>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoStringResponse xmlns:sb="http://soapinterop.org/"
        xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="xsd:string">Hello world</return>
    </sb:echoStringResponse>
  </env:Body>
</env:Envelope>
```

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"/>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
```

```
        <env:Code>
          <env:Value xmlns:ns1="http://www.w3.org/2003/05/soap-envelope">
            ns:Sender
          </env:Value>
          <env:Subcode>
            <env:Value xmlns:ns2="http://www.w3.org/2003/05/soap-rpc">
              ns:BadArguments
            </env:Value>
          </env:Subcode>
        </env:Code>
        <env:Reason>
          <env:Text xml:lang="en">Missing parameter.</env:Text>
        </env:Reason>
      </env:Fault>
    </env:Body>
  </env:Envelope>
```

Test:XMLP-2

Description:

Node A sends an HTTP GET request to Node C and Node C returns a response that includes current time. An example of the request is:

GET /interop/doc/getTime HTTP/1.1
Node C responds with the current time of the day. The response does not follow the RPC representation as described in SOAP 1.2, part 2, section 4.

Messages:

Message sent from Node A

```
  GET /soap1.2/doc/interop HTTP/1.1
  Host: www.w3.org
```

Message sent from Node C

```
  <?xml version="1.0" ?>
  <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
      <sb:time xmlns:sb="http://soapinterop.org/">09:21:19Z</sb:time>
    </env:Body>
  </env:Envelope>
```

Test:XMLP-3

Description:

The Node A sends an HTTP GET request to Node C and Node C returns an RPC response that includes current time. An example of the request is:

GET /interop/rpc/getTime HTTP/1.1
Node C responds with the current time of the day. The response follows the RPC representation as described in SOAP 1.2, part 2, section 4.

Messages:

Message sent from Node A

```
GET /soap1.2/rpc/interop HTTP/1.1
Host: www.w3.org
```

Message sent from Node C

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <sb:getTimeResponse xmlns:sb="http://soapinterop.org/"
        xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
        <rpc:result>return</rpc:result>
      <return>16:21:59Z</return>
    </sb:getTimeResponse>
  </env:Body>
</env:Envelope>
```

Test:XMLP-4

Description:

The test is based on SOAPBuilders Round 2 echoSimpleTypesAsStruct. But Node A instead of sending a message consisting of three parameters:

inputString of type xsd:string
inputInteger of type xsd:int
inputFloat of type xsd:float

in that order, it changes the order of the parameter to:

inputFloat of type xsd:float
inputInteger of type xsd:int
inputString of type xsd:string

The result returned by Node C is a structure as described in the original test.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoSimpleTypesAsStruct xmlns:sb="http://soapinterop.org/"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputFloat xsi:type="xsd:float">0.005</inputFloat>
      <inputInt xsi:type="xsd:int">42</inputInt>
      <inputString xsi:type="xsd:string">hello world</inputString>
    </sb:echoSimpleTypesAsStruct>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoSimpleTypesAsStructResponse xmlns:sb="http://soapinterop.org/"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="ns1:SOAPStruct"
              xmlns:ns1="http://soapinterop.org/xsd">
        <varString xsi:type="xsd:string">hello world</varString>
        <varInt xsi:type="xsd:int">42</varInt>
        <varFloat xsi:type="xsd:float">0.005</varFloat>
      </return>
    </sb:echoSimpleTypesAsStructResponse>
  </env:Body>
</env:Envelope>
```

Test:XMLP-5

Description:

The test is based on SOAPBuilders Round 1 echoVoid test. Node A instead of sending a message using the SOAP 1.2 namespace uses SOAP 1.1 namespace. I.e., sends the same message used in SOAPBuilders Round 2 echoVoid test. The node C returns a fault with a value of env:VersionMismatch for Code and an HTTP status code with a value of 500 Server Error.

Messages:

Message sent from Node A

```
POST /soap1.2/interop HTTP/1.1
Host: www.w3.org
Content-Type: text/xml; charset=utf-8
Content-Length: nnn
SOAPAction: ""
```

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
              xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/">
  <env:Body env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <sb:echoVoid xmlns:sb="http://soapinterop.org/" />
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <env:Upgrade>
      <env:SupportedEnvelope qname="ns1:Envelope"
                    xmlns:ns1="http://www.w3.org/2003/05/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-US">Wrong Version</env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:XMLP-6

Description:

This test is based on SOAPBuilders Round 2 echoMeUnknown test. Node A sends a request to a Node C which includes a header block with the QName {http://example.org/}Unknown and with env:mustUnderstand attribute set to "true". This header block is not understood by Node C and Node C returns a fault with a value of env:MustUnderstand for Code and an HTTP status code with a value of 500 Server Error.

Messages:

Message sent from Node A

```
POST /soap1.2/interop HTTP/1.1
Host: www.w3.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnn

<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
```

```
    <env:Header>
      <h:Unknown xmlns:h="http://example.org/"
           env:mustUnderstand="1"
           env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
           nobody understands me!
      </h:Unknown>
    </env:Header>
    <env:Body>
      <sb:echoVoid xmlns:sb="http://soapinterop.org/"
          env:encodingstyle="http://www.w3.org/2003/05/soap-encoding" />
    </env:Body>
  </env:Envelope>
```

Message sent from Node C

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <env:NotUnderstood qname="h:Unknown"
         xmlns:h="http://example.org/"/>
  </env:Header>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:MustUnderstand</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US"> Header not understood </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:XMLP-7

Description:

The Node A sends a echoSenderFault method request to the Node C and the Node C returns a fault with a value env:Sender for Code and an HTTP status code with a value of 400 Bad Method. An example of the request is:

Messages:

Message sent from Node A

```
POST /soap1.2/interop HTTP/1.1
Host: www.w3.org
Content-Type: application/soap+xml; charset=utf-8; action=""
Content-Length: nnn

<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
```

```
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <sb:echoSenderFault xmlns:sb="http://soapinterop.org/"
        xsi:type="xsd:string">
      foo
    </sb:echoSenderFault>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
HTTP/1.1 400 Bad Request
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en">This is a Sender fault.</env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:XMLP-8

Description:

The Node A sends a echoReceiverFault method request to the Node C and the Node C returns a fault with a value env:Receiver for Code and an HTTP status code with a value of 500 Server Error.

Messages:

Message sent from Node A

```
POST /soap1.2/interop HTTP/1.1
Host: www.w3.org
Content-Type: application/soap+xml; charset=utf-8; action=""
Content-Length: nnn

<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <sb:echoReceiverFault xmlns:sb="http://soapinterop.org/"
        xsi:type="xsd:string">
      foo
    </sb:echoReceiverFault>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnn

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Receiver</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en">This is a Receiver fault.</env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:XMLP-9

Description:

This test is based on SOAPBuilders Round 1 echoString test. Node A sends a request to Node C with data encoding that is unknown to Node C.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString env:encodingStyle="unknown">Hello world</inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:DataEncodingUnknown</env:value>
       </env:Code>
```

```
          <env:Reason>
            <env:Text xml:lang="en-US"> Unknown Data Encoding Style </env:Text>
          </env:Reason>
        </env:Fault>
    </env:Body>
</env:Envelope>
```

Test:XMLP-10

Description:

Node A sends a RPC message to Node C calling the method echoSimpleTypesAsStructOfSchemaTypes containing three parameters of type xsd:int, xsd:float, xsd:string, with the XML schema type information included in the message. In addtion, there is a fourth parameter which does not have the XML schema type information in the message. Node C responds with the XML schema type information corresponding to each parameter. For parameter that does not have the type information in the message, xsd:anyType is returned.

Messages:

Message sent from Node A

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoSimpleTypesAsStructOfSchemaTypes
          xmlns:test="http://soapinterop.org/ts-tests"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <input1 xsi:type="xsd:int">42</input1>
      <input2 xsi:type="xsd:float">0.005</input2>
      <input3 xsi:type="xsd:string">hello world</input3>
      <input4>Untyped information</input4>
    </test:echoSimpleTypesAsStructOfSchemaTypes>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <test:echoSimpleTypesAsStructOfSchemaTypesResponse
          xmlns:test="http://soapinterop.org/ts-tests"
          xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>return</rpc:result>
      <return xsi:type="ns1:SOAPStructTypes"
              xmlns:ns1="http://example.org/ts-tests/xsd">
        <type1 xsi:type="xsd:QName">xsd:int</type1>
        <type2 xsi:type="xsd:QName">xsd:float</type2>
```

```
          <type3 xsi:type="xsd:QName">xsd:string</type3>
          <type4 xsi:type="xsd:QName">xsd:anyType</type4>
        </return>
      </test:echoSimpleTypesAsStructOfSchemaTypesResponse>
    </env:Body>
  </env:Envelope>
```

Test:XMLP-11

Description:

This test is based on SOAPBuilders R1 echoInteger test. But instead of sending a valid integer value, Node A sends a string value containing non-interger characters. The Node C returns a fault with a value env:Sender for Code, a value of rpc:BadArguments for Subcode.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoInteger xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputInteger xsi:type="xsd:int">abc</inputInteger>
    </sb:echoInteger>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
        <env:Subcode>
          <env:Value xmlns:rpc="http://www.w3.org/2003/05/soap-rpc">
            rpc:BadArguments
          </env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en">
          Bad parameter: 'inputInteger' on method 'echoInteger'.
        </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:XMLP-12

Description:

This test is based on SOAPBuilders R1 echoInteger test. But instead of of sending an echoInteger RPC request Node A sends a RPC request for the method unknownMethodThatShouldNotBeThere. The Node C returns a fault with a value env:Sender for Code, a value of rpc:ProcedureNotPresent for Subcode.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:unknownMethodThatShouldNotBeThere xmlns:sb="http://soapinterop.org/" />
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:rpc="http://www.w3.org/2003/05/soap-rpc">
  <env:Body>
     <env:Fault>
       <env:Code>
         <env:Value>env:Sender</env:Value>
         <env:Subcode>
           <env:Value>rpc:ProcedureNotPresent</env:Value>
         </env:Subcode>
       </env:Code>
       <env:Reason>
        <env:Text xml:lang="en-US"> Procedure Not Present </env:Text>
       </env:Reason>
     </env:Fault>
  </env:Body>
</env:Envelope>
```

Test:XMLP-13

Description:

The test is based on SOAPBuilders R1 echoString test. But, instead of testing RPC request/response, it tests the SOAP 1.2 forwarding intermediary semantics. Node A sends a echoString request and Node C acts as a forwarding intermediary. But, instead of forwarding the message to another node, it forwards the message back to Node A.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">Hello world</inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">Hello world</inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Test:XMLP-14

Description:

The test is based on SOAPBuilders R1 echoString test. But, instead of testing RPC request/response, it tests the SOAP 1.2 active intermediary semantics. Node A sends a echoString request and Node C acts as a active intermediary. But, instead of forwarding the message to another node, it forwards the message back to Node A. Node C in addition to forwarding the message to Node A, converts the string to be echoed to uppercase before forwarding it to Node A.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">Hello world</inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">HELLO WORLD<inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Test:XMLP-15

Description:

The test is similar to XMLP-13. Node A sends the echoString request along with an optional SOAP header block targeted at "http://www.w3.org/2003/05/soap-envelope/role/next" Node C does not process the header block, but removes the header block before forwarding the message back to Node A.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Header>
    <sb:Unknown env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
                xmlns:sb="http://soapinterop.org/">
    </sb:Unknown>
  </env:Header>
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">Hello world<inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">Hello world</inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Test:XMLP-16

Description:

The test is similar to XMLP-13. Node A sends the echoString request along with an optional SOAP header block targeted at "http://www.w3.org/2003/05/soap-envelope/role/none" Node C does not process the header block, but retains the header block in the message forwarded back to Node A.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Header>
    <sb:Unknown env:role="http://www.w3.org/2003/05/soap-envelope/role/none"
                xmlns:sb="http://soapinterop.org/">
    </sb:Unknown>
  </env:Header>
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">Hello world</inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Header>
    <sb:Unknown env:role="http://www.w3.org/2003/05/soap-envelope/role/none"
                xmlns:sb="http://soapinterop.org/">
    </sb:Unknown>
  </env:Header>
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">Hello world</inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Test:XMLP-17

Description:

The test is similar to XMLP-13. Node A sends the echoString request along with an optional SOAP header block targeted at "http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver" Node C does not process the header block, but retains the header block in the message forwarded back to Node A.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Header>
    <sb:Unknown
        env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
        xmlns:sb="http://soapinterop.org/">
    </sb:Unknown>
  </env:Header>
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">Hello world</inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Header>
    <sb:Unknown
        env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
        xmlns:sb="http://soapinterop.org/">
    </sb:Unknown>
  </env:Header>
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">Hello world</inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Test:XMLP-18

Description:

The test is similar to XMLP-15. Node A sends the echoString request along with an optional SOAP header block targeted at "http://www.w3.org/2003/05/soap-envelope/role/next" and a relay attribute with a value of "true". Node C does not process the header block, but retains the header block in the message forwarded back to Node A (along with the role and relay attribute).

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Header>
    <sb:Unknown
        env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
        env:relay="true"
        xmlns:sb="http://soapinterop.org/">
    </sb:Unknown>
  </env:Header>
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">Hello world</inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Header>
    <sb:Unknown
        env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
        env:relay="true"
        xmlns:sb="http://soapinterop.org/">
    </sb:Unknown>
  </env:Header>
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">Hello world</inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Test:XMLP-19

Description:

The test is similar to XMLP-15. Node A sends the echoString request along with an SOAP header block targeted at "http://www.w3.org/2003/05/soap-envelope/role/next" and a mustUnderstand attribute with a value of "true". Node C does not recognize the header block, and this results in a MustUnderstand fault which is sent back to Node A.

Messages:

Message sent from Node A

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <env:Header>
    <sb:Unknown
        env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
        env:mustUnderstand="true"
        xmlns:sb="http://soapinterop.org/">
    </sb:Unknown>
  </env:Header>
  <env:Body>
    <sb:echoString xmlns:sb="http://soapinterop.org/"
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <inputString xsi:type="xsd:string">Hello world</inputString>
    </sb:echoString>
  </env:Body>
</env:Envelope>
```

Message sent from Node C

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <env:NotUnderstood qname="sb:Unknown"
         xmlns:sb="http://soapinterop.org/" />
  </env:Header>
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:MustUnderstand</env:Value>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US"> Header not understood </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

4. References

4.1 Normative References

[SOAP Part 1]
SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, Yves Lafon, Editors. World Wide Web Consortium, 27 April 2007. This version is http://www.w3.org/TR/2007/REC-soap12-part1-20070427. The latest version is available at http://www.w3.org/TR/soap12-part1/.

[SOAP Part 2]
SOAP Version 1.2 Part 2: Adjuncts (Second Edition), Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, Yves Lafon, Editors. World Wide Web Consortium, 27 April 2007. This version is http://www.w3.org/TR/2007/REC-soap12-part2-20070427. The latest version is available at http://www.w3.org/TR/soap12-part2/.

[XML 1.0]
Extensible Markup Language (XML) 1.0 (Fourth Edition), Jean Paoli, Eve Maler, Tim Bray, *et. al.*, Editors. World Wide Web Consortium, 16 August 2006. This version is http://www.w3.org/TR/2006/REC-xml-20060816. The latest version is available at

http://www.w3.org/TR/REC-xml.

[XML Schema Part 1]
XML Schema Part 1: Structures Second Edition, David Beech, Murray Maloney, Henry S. Thompson, and Noah Mendelsohn, Editors. World Wide Web Consortium, 28 October 2004. This version is http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/ The latest version is available at http://www.w3.org/TR/xmlschema-1/.

[XML Schema Part 2]
XML Schema Part 2: Datatypes Second Edition, Ashok Malhotra and Paul V. Biron, Editors. World Wide Web Consortium, 28 October 2004. This version is http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/. The latest version is available at http://www.w3.org/TR/xmlschema-2/.

4.2 Informative References

[distapp archive]
XML Protocol Discussion Archive (See http://lists.w3.org/Archives/Public/xml-dist-app/.)

[xmlp-comments archive]
XML Protocol Comments Archive (See http://lists.w3.org/Archives/Public/xmlp-comments/.)

[SOAPBuilders Round 1]
SOAPBuilders Round 1 SOAP Interoperability Tests Specification (See http://www.xmethods.net/soapbuilders/proposal.html.)

[SOAPBuilders Round 2]
SOAPBuilders Round 2 SOAP Interoperability Tests Specification (See http://www.whitemesa.com/interop/proposal2.html.)

[SOAPBuilders Round 2 Group B]
SOAPBuilders Round 2 SOAP Interoperability Tests Specification: Group "B" Methods (See http://www.whitemesa.com/interop/proposalB.html.)

[SOAPBuilders Round 2 Group C]
SOAPBuilders Round 2 SOAP Interoperability Tests Specification: Header Processing (See http://www.whitemesa.com/interop/proposalC.html.)

A. Acknowledgements (Non-Normative)

This specification is the work of the W3C XML Protocol Working Group.

Members of the Working Group are (at the time of writing, and by alphabetical order): Glen Daniels (Sonic Software, formerly of Macromedia), Vikas Deolaliker (Sonoa Systems, Inc.), Chris Ferris (IBM, formerly of Sun Microsystems), Marc Hadley (Sun Microsystems), David Hull (TIBCO Software, Inc.), Anish Karmarkar (Oracle), Yves Lafon (W3C), Jonathan Marsh (WSO2), Jeff Mischkinsky (Oracle), Eric Newcomer (IONA Technologies), David Orchard (BEA Systems, formerly of Jamcracker), Seumas Soltysik (IONA Technologies), Davanum Srinivas (WSO2), Pete Wenzel (Sun Microsystems, formerly of SeeBeyond).

Previous members were: Yasser alSafadi (Philips Research), Bill Anderson (Xerox), Vidur Apparao (Netscape), Camilo Arbelaez (webMethods), Mark Baker (Idokorro Mobile, Inc., formerly of Sun Microsystems), Philippe Bedu (EDF (Electricite De France)), Olivier Boudeville (EDF (Electricite De France)), Carine Bournez (W3C), Don Box (Microsoft Corporation, formerly of DevelopMentor), Tom Breuel (Xerox), Dick Brooks (Group 8760), Winston Bumpus (Novell, Inc.), David Burdett (Commerce One), Charles Campbell (Informix Software), Alex Ceponkus (Bowstreet), Michael Champion (Software AG), David Chappell (Sonic Software), Miles Chaston (Epicentric), David Clay (Oracle), David Cleary (Progress Software), Dave Cleary (webMethods), Ugo Corda (Xerox), Paul Cotton (Microsoft

Corporation), Fransisco Cubera (IBM), Jim d'Augustine (Excelon Corporation), Ron Daniel (Interwoven), Doug Davis (IBM), Ray Denenberg (Library of Congress), Paul Denning (MITRE Corporation), Frank DeRose (TIBCO Software, Inc.), Mike Dierken (DataChannel), Andrew Eisenberg (Progress Software), Brian Eisenberg (DataChannel), Colleen Evans (Sonic Software), John Evdemon (XMLSolutions), David Ezell (Hewlett Packard), James Falek (TIBCO Software, Inc.), David Fallside (IBM), Eric Fedok (Active Data Exchange), Daniela Florescu (Propel), Dan Frantz (BEA Systems), Michael Freeman (Engenia Software), Dietmar Gaertner (Software AG), Scott Golubock (Epicentric), Tony Graham (Sun Microsystems), Mike Greenberg (IONA Technologies), Rich Greenfield (Library of Congress), Martin Gudgin (Microsoft Corporation, formerly of DevelopMentor), Hugo Haas (W3C), Mark Hale (Interwoven), Randy Hall (Intel), Bjoern Heckel (Epicentric), Frederick Hirsch (Zolera Systems), Gerd Hoelzing (SAP AG), Erin Hoffmann (Tradia Inc.), Steve Hole (MessagingDirect Ltd.), Mary Holstege (Calico Commerce), Jim Hughes (Fujitsu Limited), Oisin Hurley (IONA Technologies), Yin-Leng Husband (Hewlett Packard, formerly of Compaq), John Ibbotson (IBM), Ryuji Inoue (Matsushita Electric Industrial Co., Ltd.), Scott Isaacson (Novell, Inc.), Kazunori Iwasa (Fujitsu Limited), Murali Janakiraman (Rogue Wave), Mario Jeckle (DaimlerChrysler Research and Technology), Eric Jenkins (Engenia Software), Mark Jones (AT&T), Jay Kasi (Commerce One), Jeffrey Kay (Engenia Software), Suresh Kodichath (IONA Technologies), Richard Koo (Vitria Technology Inc.), Jacek Kopecky (Systinet), Alan Kropp (Epicentric), Julian Kumar (Epicentric), Peter Lecuyer (Progress Software), Tony Lee (Vitria Technology Inc.), Michah Lerner (AT&T), Bob Lojek (Intalio Inc.), Henry Lowe (OMG), Brad Lund (Intel), Matthew MacKenzie (XMLGlobal Technologies), Michael Mahan (Nokia), Murray Maloney (Commerce One), Richard Martin (Active Data Exchange), Noah Mendelsohn (IBM, formerly of Lotus Development), Alex Milowski (Lexica), Kevin Mitchell (XMLSolutions), Nilo Mitra (Ericsson), Ed Mooney (Sun Microsystems), Jean-Jacques Moreau (Canon), Dean Moses (Epicentric), Highland Mary Mountain (Intel), Don Mullen (TIBCO Software, Inc.), Rekha Nagarajan (Calico Commerce), Raj Nair (Cisco Systems), Masahiko Narita (Fujitsu Limited), Mark Needleman (Data Research Associates), Art Nevarez (Novell, Inc.), Henrik Nielsen (Microsoft Corporation), Mark Nottingham (BEA Systems, formerly of Akamai Technologies), Conleth O'Connell (Vignette), Kevin Perkins (Compaq), Doug Purdy (Microsoft Corporation), Jags Ramnaryan (BEA Systems), Andreas Riegg (DaimlerChrysler Research and Technology), Vilhelm Rosenqvist (NCR), Herve Ruellan (Canon), Marwan Sabbouh (MITRE Corporation), Waqar Sadiq (Vitria Technology Inc.), Rich Salz (Zolera Systems), Krishna Sankar (Cisco Systems), Jeff Schlimmer (Microsoft Corporation), George Scott (Tradia Inc.), Shane Sesta (Active Data Exchange), Lew Shannon (NCR), John-Paul Sicotte (MessagingDirect Ltd.), Miroslav Simek (Systinet), Simeon Simeonov (Macromedia), Aaron Skonnard (DevelopMentor), Nick Smilonich (Unisys), Soumitro Tagore (Informix Software), James Tauber (Bowstreet), Anne Thomas Manes (Sun Microsystems), Lynne Thompson (Unisys), Patrick Thompson (Rogue Wave), Jim Trezzo (Oracle), Asir Vedamuthu (webMethods), Mike Vernal (Microsoft Corporation), Randy Waldrop (WebMethods), Fred Waskiewicz (OMG), David Webber (XMLGlobal Technologies), Ray Whitmer (Netscape), Volker Wiechers (SAP AG), Stuart Williams (Hewlett Packard), Yan Xu (DataChannel), Amr Yassin (Philips Research), Susan Yee (Active Data Exchange), Jin Yu (MartSoft Corp.).

The people who have contributed to discussions on xml-dist-app@w3.org are also gratefully acknowledged.

The editors would like to acknowledge Kirill Gavrylyuk (Microsoft Corp.) for the gladly-received contribution of test for SOAP 1.2 Part 1.

The editors would like to acknowledge Nick Smilonich for reviewing this document.