

SOAP with Attachments API for Java™ (SAAJ) 1.4

Maintenance Release

Editors:

Lukas Jungmann

Oracle Corporation
www.oracle.com

July 2017, Revision 01

Submit comments about this document to: javaee-spec@javaee.groups.io

ORACLE AMERICA, INC. IS WILLING TO LICENSE THIS SPECIFICATION TO YOU ONLY UPON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN THIS AGREEMENT. PLEASE READ THE TERMS AND CONDITIONS OF THIS AGREEMENT CAREFULLY. BY DOWNLOADING THIS SPECIFICATION, YOU ACCEPT THE TERMS AND CONDITIONS OF THE AGREEMENT. IF YOU ARE NOT WILLING TO BE BOUND BY IT, SELECT THE "DECLINE" BUTTON AT THE BOTTOM OF THIS PAGE.

Specification: JSR – 67 SOAP with Attachments API for Java ("Specification")

Version: 1.4

Status: Maintenance Release

Specification Lead: Oracle America, Inc. ("Specification Lead")

Release: July 2017

Copyright 2017 Oracle America, Inc. All rights reserved.

LIMITED LICENSE GRANTS

1. License for Evaluation Purposes. Specification Lead hereby grants you a fully-paid, non-exclusive, non-transferable, worldwide, limited license (without the right to sublicense), under Specification Lead's applicable intellectual property rights to view, download, use and reproduce the Specification only for the purpose of internal evaluation. This includes (i) developing applications intended to run on an implementation of the Specification, provided that such applications do not themselves implement any portion(s) of the Specification, and (ii) discussing the Specification with any third party; and (iii) excerpting brief portions of the Specification in oral or written communications which discuss the Specification provided that such excerpts do not in the aggregate constitute a significant portion of the Specification.

2. License for the Distribution of Compliant Implementations. Specification Lead also grants you a perpetual, non-exclusive, non-transferable, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights or, subject to the provisions of subsection 4 below, patent rights it may have covering the Specification to create and/or distribute an Independent Implementation of the Specification that: (a) fully implements the Specification including all its required interfaces and functionality; (b) does not modify, subset, superset or otherwise extend the Licensor Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the Licensor Name Space other than those required/authorized by the Specification or Specifications being implemented; and (c) passes the Technology Compatibility Kit (including satisfying the requirements of the applicable TCK Users Guide) for such Specification ("Compliant Implementation"). In addition, the foregoing license is expressly conditioned on your not acting outside its scope. No license is granted hereunder for any other purpose (including, for example, modifying the Specification, other than to the extent of your fair use rights, or

distributing the Specification to third parties). Also, no right, title, or interest in or to any trademarks, service marks, or trade names of Specification Lead or Specification Lead's licensors is granted hereunder. Java, and Java-related logos, marks and names are trademarks or registered trademarks of Oracle America, Inc. in the U.S. and other countries.

3. Pass-through Conditions. You need not include limitations (a)-(c) from the previous paragraph or any other particular "pass through" requirements in any license You grant concerning the use of your Independent Implementation or products derived from it. However, except with respect to Independent Implementations (and products derived from them) that satisfy limitations (a)-(c) from the previous paragraph, You may neither: (a) grant or otherwise pass through to your licensees any licenses under Specification Lead's applicable intellectual property rights; nor (b) authorize your licensees to make any claims concerning their implementation's compliance with the Specification in question.

4. Reciprocity Concerning Patent Licenses.

a. With respect to any patent claims covered by the license granted under subparagraph 2 above that would be infringed by all technically feasible implementations of the Specification, such license is conditioned upon your offering on fair, reasonable and non-discriminatory terms, to any party seeking it from You, a perpetual, non-exclusive, non-transferable, worldwide license under Your patent rights which are or would be infringed by all technically feasible implementations of the Specification to develop, distribute and use a Compliant Implementation.

b With respect to any patent claims owned by Specification Lead and covered by the license granted under subparagraph 2, whether or not their infringement can be avoided in a technically feasible manner when implementing the Specification, such license shall terminate with respect to such claims if You initiate a claim against Specification Lead that it has, in the course of performing its responsibilities as the Specification Lead, induced any other entity to infringe Your patent rights.

c Also with respect to any patent claims owned by Specification Lead and covered by the license granted under subparagraph 2 above, where the infringement of such claims can be avoided in a technically feasible manner when implementing the Specification such license, with respect to such claims, shall terminate if You initiate a claim against Specification Lead that its making, having made, using, offering to sell, selling or importing a Compliant Implementation infringes Your patent rights.

5. Definitions. For the purposes of this Agreement: "Independent Implementation" shall mean an implementation of the Specification that neither derives from any of Specification Lead's source code or binary code materials nor, except with an appropriate and separate license from Specification Lead, includes any of Specification Lead's source code or binary code materials; "Licensor Name Space" shall mean the public class or interface declarations whose names begin with "java", "javax", "com.oracle", "com.sun" or their equivalents in any subsequent naming convention adopted by Oracle America, Inc. through the Java Community Process, or any recognized successors or replacements thereof; and "Technology Compatibility Kit" or "TCK" shall mean the test suite and accompanying TCK User's Guide provided by Specification Lead which corresponds to the Specification and that was

available either (i) from Specification Lead 120 days before the first release of Your Independent Implementation that allows its use for commercial purposes, or (ii) more recently than 120 days from such release but against which You elect to test Your implementation of the Specification.

This Agreement will terminate immediately without notice from Specification Lead if you breach the Agreement or act outside the scope of the licenses granted above.

DISCLAIMER OF WARRANTIES

THE SPECIFICATION IS PROVIDED "AS IS". SPECIFICATION LEAD MAKES NO REPRESENTATIONS OR WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT (INCLUDING AS A CONSEQUENCE OF ANY PRACTICE OR IMPLEMENTATION OF THE SPECIFICATION), OR THAT THE CONTENTS OF THE SPECIFICATION ARE SUITABLE FOR ANY PURPOSE. This document does not represent any commitment to release or implement any portion of the Specification in any product. In addition, the Specification could include technical inaccuracies or typographical errors.

LIMITATION OF LIABILITY

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SPECIFICATION LEAD OR ITS LICENSORS BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION, LOST REVENUE, PROFITS OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED IN ANY WAY TO YOUR HAVING, IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION, EVEN IF SPECIFICATION LEAD AND/OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. You will indemnify, hold harmless, and defend Specification Lead and its licensors from any claims arising or resulting from: (i) your use of the Specification; (ii) the use or distribution of your Java application, applet and/or implementation; and/or (iii) any claims that later versions or releases of any Specification furnished to you are incompatible with the Specification provided to you under this license.

RESTRICTED RIGHTS LEGEND

U.S. Government: If this Specification is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in the Software and accompanying documentation shall be only as set forth in this license; this is in accordance with 48 C.F.R. 227.7201 through 227.7202-4 (for Department of Defense (DoD) acquisitions) and with 48 C.F.R. 2.101 and 12.212 (for non-DoD acquisitions).

REPORT

If you provide Specification Lead with any comments or suggestions concerning the Specification ("Feedback"), you hereby: (i) agree that such Feedback is provided on a non-proprietary and non-confidential basis, and (ii) grant Specification Lead a perpetual, non-exclusive, worldwide, fully paid-up, irrevocable license, with the right to sublicense through multiple levels of sublicensees, to incorporate, disclose, and use without limitation the Feedback for any purpose.

GENERAL TERMS

Any action related to this Agreement will be governed by California law and controlling U.S. federal law. The U.N. Convention for the International Sale of Goods and the choice of law rules of any jurisdiction will not apply.

The Specification is subject to U.S. export control laws and may be subject to export or import regulations in other countries. Licensee agrees to comply strictly with all such laws and regulations and acknowledges that it has the responsibility to obtain such licenses to export, re-export or import as may be required after delivery to Licensee.

This Agreement is the parties' entire agreement relating to its subject matter. It supersedes all prior or contemporaneous oral or written communications, proposals, conditions, representations and warranties and prevails over any conflicting or additional terms of any quote, order, acknowledgment, or other communication between the parties relating to its subject matter during the term of this Agreement. No modification to this Agreement will be binding, unless in writing and signed by an authorized representative of each party.

Contents

Contents ii

Status vi

Status of This Document vi

Acknowledgements vi

Terminology vii

Preface viii

Audience viii

Abstract viii

Change History ix

Typographic Conventions xi

1. Package Overview 1-1

1.1 MessageFactory & SOAPMessage Objects 1-1

1.2 SOAPPart & AttachmentPart 1-2

1.3 MimeHeader(s) Objects 1-3

1.4 SOAP Element 1-3

1.5 SOAPEnvelope & SOAPBody objects 1-3

1.6 SOAPBodyElement & SOAPFault 1-4

1.7 SOAPFaultElement & Detail 1-4

- 1.8 SOAPHeader & SOAPHeaderElement 1–4
- 1.9 SOAPConnection & SOAPConnectionFactory 1–5
- 1.10 SOAPException object 1–5
- 1.11 Node & Text objects 1–5
- 1.12 Name 1–6
- 1.13 SOAPFactory & SOAPElementFactory 1–6
- 1.14 SAAJMetaFactory 1–6
- 1.15 SAAJResult 1–6

2. Package: javax.xml.soap 2–1

- 2.1 Description 2–1
- 2.2 AttachmentPart 2–5
- 2.3 Detail 2–17
- 2.4 DetailEntry 2–20
- 2.5 MessageFactory 2–22
- 2.6 MimeHeader 2–26
- 2.7 MimeHeaders 2–28
- 2.8 Name 2–32
- 2.9 Node 2–35
- 2.10 SAAJMetaFactory 2–39
- 2.11 SAAJResult 2–42
- 2.12 SOAPBody 2–46
- 2.13 SOAPBodyElement 2–54
- 2.14 SOAPConnection 2–56
- 2.15 SOAPConnectionFactory 2–59
- 2.16 SOAPConstants 2–61
- 2.17 SOAPElement 2–67
- 2.18 SOAPElementFactory 2–81
- 2.19 SOAPEnvelope 2–84

2.20	SOAPException	2–89
2.21	SOAPFactory	2–93
2.22	SOAPFault	2–100
2.23	SOAPFaultElement	2–113
2.24	SOAPHeader	2–115
2.25	SOAPHeaderElement	2–123
2.26	SOAPMessage	2–128
2.27	SOAPPart	2–139
2.28	Text	2–146

A. References A–1

Status

Status of This Document

This specification was developed following the Java™ Community Process (JCP2.10). Comments from experts, participants, and the broader developer community were reviewed and incorporated into this specification.

The SAAJ Specification, version 1.1 was a maintenance release of the Java™ API for XML Messaging (JAXM) 1.0 specification. JAXM 1.0 was the final deliverable of JSR067 Expert Group (EG). The proposed changes specified in the JSR067 changelog and accepted on 15 April 2002, have been incorporated into this document.

The proposed changes specified in the second JSR067 changelog and accepted on 23 April 2003, have been incorporated into this document as SAAJ Specification, version 1.2.

The proposed changes specified in the third JSR067 changelog have been incorporated into this document as SAAJ Specification, version 1.3.

The proposed changes specified in the fourth JSR067 changelog have been incorporated into this document as SAAJ Specification, version 1.4.

Acknowledgements

This maintenance release is the product of collaborative work within the Java community.

Terminology

The keywords **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL**, when they appear in this document, are to be interpreted as described in RFC 2119 as quoted here:

MUST: This word, or the terms “**REQUIRED**” or “**SHALL**”, mean that the definition is an absolute requirement of the specification.

MUST NOT: This phrase, or the phrase “**SHALL NOT**”, mean that the definition is an absolute prohibition of the specification.

SHOULD: This word, or the adjective “**RECOMMENDED**”, mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

SHOULD NOT: This phrase, or the phrase “**NOT RECOMMENDED**”, mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

MAY: This word, or the adjective “**OPTIONAL**”, mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option **MUST** be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option **MUST** be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides).

Preface

Audience

This document is intended for developers using the Java™ programming language who wish to produce and consume messages conforming to the SOAP 1.1₂ and SOAP 1.2 specification and the SOAP Attachments Feature.

Familiarity with the SOAP specifications (including the associated processing model), MIME standards, and XML is assumed.

Abstract

The SOAP with Attachments API for Java™ (SAAJ 1.4) enables developers to produce and consume messages conforming to the SOAP 1.1, SOAP 1.2 and SOAP Attachments Feature.

In the interest of backward compatibility, SAAJ continues to offer a client-side communication capability enabling developers to communicate in a point-to-point and request-response manner with SOAP services bound to HTTP. This communication capability, within the context of the SAAJ specification, is optional. However, specifications depending on SAAJ are free to require support for the SOAP to HTTP binding.

Change History

Changes in Maintenance Release 4

The fourth maintenance release of SAAJ, SAAJ 1.4, clarifies and extends the SAAJ 1.3 specification. The goal of this maintenance release proposal is primarily to adopt changes in Java SE Platform and to make a few corrections and clarifications on the existing SAAJ 1.3 APIs. The proposed API changes in SAAJ 1.4 are backward compatible with SAAJ 1.3 APIs.

The “accepted changes,” as specified in JSR067 changelog, have been incorporated in this document. The key changes are as follows:

- Changed the definition of the discovery process of SAAJ implementation.
- Added support for generics where applicable.
- Changed MAY to MUST in `javax.xml.soap.SOAPElements` methods.
- Several corrections and clarifications have been made to the JavaDocs for the API.

Changes in Maintenance Release 3

The third maintenance release of SAAJ, SAAJ 1.3, clarifies and extends the SAAJ 1.2 specification. The goal of this maintenance release proposal is primarily to provide support for SOAP version 1.2 Message Constructs and to make a few corrections and clarifications on the existing SAAJ 1.2 APIs. The proposed API changes in SAAJ 1.3 are backward compatible with SAAJ 1.2 APIs.

SOAP version 1.2 has a number of changes in syntax and provides additional (or clarified) semantics from those described in SOAP 1.1. This proposed changes in this maintenance release are concerned with the following areas:

- Support for SOAP version 1.2 message constructs in the API.
- Factoring out the creation of all SAAJ Factory classes into a single SPI that allows creation of SOAP version aware Factories.
- Addition of a few new classes and new methods in certain existing classes and interfaces.
- Support for overloaded QName based methods in certain classes and interfaces.
- Clarification of semantics and correction of wording of JavaDocs and specification

A brief summary of the proposed changes follows:

- Support for SOAP Version 1.2 message constructs in the API: SOAP Version 1.2 has a number of changes in syntax and introduces several new Message Constructs. SAAJ 1.3 will support SOAP Version 1.2 Message Constructs.
- SPI for Creation of Factory Instances: SAAJ 1.3 will support SOAP Version 1.2 Message Constructs, while at the same time being backward compatible in its support for SOAP Version 1.1. We would like to define an SPI (SAAJMetaFactory) for factoring out the creation of SOAP Version aware Factory classes into a single place. Changing out the SAAJMetaFactory has the effect of changing out the entire SAAJ implementation. Backward compatibility is maintained by ensuring that the default protocol is set to SOAP Version 1.1.
- Definition of new Class SAAJResult: A SAAJResult object acts as a holder for the results of a JAXP transformation or a JAXB marshalling, in the form of a SAAJ tree. This class will make it easier for the end user when dealing with transformations in situations where the result is expected to be a valid SAAJ tree.
- Addition of overloaded methods which accept a QName instead of a Name: QName is the preferred representation of XML qualified names, and hence we would like to introduce overloaded methods in all APIs where a corresponding method was accepting a javax.xml.soap.Name as argument. The Name interface may be deprecated in a future release of SAAJ in favor of QName.
- Clarify and correct the wording of JavaDocs and specification: None of these changes will break backward compatibility for SOAP 1.1 users. Corrections of this nature cost little and improve the overall integrity of the specification making correct implementations easier to create, validate and use.
- Addition of new methods in certain Interfaces and Classes: A few new methods have been introduced in AttachmentPart, SOAPBody, and SOAPElement. These new methods are intended for ease of use and to assist SAAJ users when dealing with some of the newer SOAP features.
- Making SOAPPart a javax.xml.soap.Node: The SOAPPart is also a SOAP Node.
- Deferred Changes: The deprecation of Name Interface has been deferred to a later release.
- DOM Level 3 Support: Implementations of SAAJ 1.3 must provide support for DOM Level 3 APIs.

Changes in Maintenance Release 2

The second maintenance release of SAAJ, SAAJ 1.2, clarifies and extends the SAAJ 1.1 specification. The “accepted changes”, as specified in the *Change Log for SOAP with Attachments API for Java™*, have been incorporated into this document. A summary of the changes follows:

- The core SAAJ classes and interfaces: Node, SOAPElement, SOAPPart, and Text now extend the equivalent interfaces in the org.w3c.dom package: Node, Element, Document and Text respectively.

- The ability to get and set properties on `SOAPMessage` has been added to `SOAPMessage` in order to facilitate extensibility and two new properties have been added in order to take advantage of this extensibility: `CHARACTER_SET_ENCODING` allows the character encoding to be set to “utf-8” or “utf-16” where “utf-8” is the default. Implementations may optionally support other character encodings. `WRITE_XML_DECLARATION` allows clients to specify whether or not an XML Declaration will be written at the start of the SOAP part of the message. The valid values are “true” and “false” with “false” being the default.
- Several APIs have been extended in order to provide greater ease of use. The `Node` interface has gained a `setValue()` method. `SOAPFault` has been enhanced with several methods that facilitate the handling of its sub-elements. `SOAPMessage`, `SOAPElement`, `SOAPBody` and `SOAPHeader` have all been given new methods that enhance navigation of the tree. A `removeContents()` element has been added to `SOAPElement` in order to assist in the construction of messages that contain a fault.
- Several corrections and clarifications have been made to the JavaDocs for the API.

This specification has been derived from the `javax.xml.soap` package originally defined in the JAXM 1.0 specification. The “accepted changes,” as specified in JSR067 changelog, have been incorporated in this document. The key changes are as follows:

- `javax.xml.soap` package was moved from the JAXM specification to this document. In the interest of consistency and for simplifying synchronization of specifications, this document has been designated as version 1.1 of the SAAJ specification. There are no prior versions of the SAAJ specification.
- The `call` method signature of the `SOAPConnection` object has been modified so as to remove the dependency of SAAJ on JAXM.
- The `newInstance` method of `SOAPConnectionFactory` may throw an `UnsupportedOperationException` hence making the implementation of the `SOAPConnection.call()` functionality optional.
- The `SOAPElementFactory` has been deprecated and a new “super” factory for creating `Element`, `Detail`, and `Name` objects created. The previous `SOAPElementFactory` methods now delegate to the appropriate `SOAPFactory` methods.

Typographic Conventions

Typeface*	Meaning	Examples
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. % You have mail.
AaBbCc123	What you type, when contrasted with on-screen computer output	% su Password:
<i>AaBbCc123</i>	Book titles, new words or terms, words to be emphasized. Replace command-line variables with real names or values.	Read Chapter 6 in the <i>User's Guide</i> . These are called <i>class</i> options. You <i>must</i> be superuser to do this. To delete a file, type <code>rm filename</code> .

* The settings on your browser might differ from these settings.

Package Overview

This chapter presents an overview of the SAAJ which consists of the single package; `javax.xml.soap`. The intent here is to provide an overview of the package only, the details of which can be found in the following chapter.

The `javax.xml.soap` package provides the primary abstraction for SOAP Messages with MIME attachments. Attachments may be entire XML documents, XML fragments, images, text documents, or any other content with a valid MIME type. In addition, this package provides a simple client-side view of a request-response style of interaction with a SOAP service.

1.1 MessageFactory & SOAPMessage Objects

The `MessageFactory` class is used to create `SOAPMessage` objects. Clients may create `SOAPMessage` objects by calling the `MessageFactory.createMessage` method.

The `SOAPMessage` class is the root class for all SOAP messages. Such messages must contain a single `SOAPPart` object and may contain one or more `AttachmentPart` objects. The “on-the-wire” encoding of a SOAP message is governed by whether the `SOAPMessage` object includes `AttachmentPart` objects. If it does, the `SOAPMessage` object is encoded as a MIME message otherwise it is encoded as a simple XML message. Attachments may contain data of any type including XML. The `SOAPPart` is always XML.

SAAJ allows for creation and consumption of both SOAP 1.1 and SOAP 1.2 messages by introducing the notion of Protocol aware `MessageFactories`. The protocol here refers to a particular version of SOAP. For example a SOAP 1.2 aware `MessageFactory` can be obtained by calling the `MessageFactory.newInstance` method and passing it the appropriate protocol identifier. The allowed protocol identifiers have been defined in `SOAPConstants`. For processing incoming messages a special protocol identifier called `DYNAMIC_SOAP_PROTOCOL` can be used to allow a Node to accept both SOAP 1.1 and SOAP 1.2 messages.

1.2 SOAPPart & AttachmentPart

The SOAPPart object is a MIME part containing the SOAPEnvelope object. The SOAPEnvelope object must contain a single SOAPBody object and may contain a SOAPHeader object.

A SOAPMessage object may contain zero or more AttachmentPart objects. Each AttachmentPart object in turn contains application-specific content and corresponding MIME headers. The MIME headers consist of name/value pairs that are used to identify and describe the content. For MIME content-types of `text/plain`, `text/html` and `text/xml`, the `DataContentHandler` object performs the necessary conversions to and from the Java types corresponding to the MIME types. Other MIME types can be supported by passing an `InputStream` object (that contains the content data) to the `AttachmentPart.setContent` method. Similarly, the contents and header from an `AttachmentPart` object can be retrieved using the `getContent` method. Depending on the `AttachmentPart` objects present, the returned `Object` can be either a typed Java object corresponding to the MIME type or an `InputStream` object that contains the content as bytes. The `clearContent` method is a helper method intended to facilitate the removal of all the content from an `AttachmentPart` object while leaving the header information.

A SAAJ 1.4 implementation must support the following MIME types. Additional MIME types may be supported using the `javax.activation.DataHandler` class and the Java™ Activation Framework.

TABLE 1-1 SAAJ 1.4's supported MIME types

MIME Type	Java Type
<code>text/plain</code>	<code>java.lang.String</code>
<code>multipart/*</code>	<code>javax.mail.internet.MimeMultipart</code>
<code>text/xml</code> or <code>application/xml</code>	<code>javax.xml.transform.Source</code>

SAAJ provides methods for setting and getting the Raw content of an Attachment. Methods have also been provided to get the content as Base64 encoded character data. Additionally a `getAttachment` method on the `SOAPMessage` provides for retrieval of an Attachment referenced from a `SOAPElement` using an `href` attribute as described in SOAP Messages with Attachments, or via a single `Text` child node containing a URI as described in the WS-I Attachments Profile 1.0 for elements of schema type `ref:swaRef`

1.3 MimeHeader(s) Objects

The `MIMEHeaders` class is a container for `MimeHeader` objects and serves as an abstraction for the MIME headers that must be present if an `AttachmentPart` object exists in a `SOAPMessage` object.

The `MimeHeader` object is the abstraction for a name/value pair of a MIME header. A `MIMEHeaders` object may contain one or more `MimeHeader` objects.

1.4 SOAP Element

The `SOAPElement` object is the base class for all of the classes that model the SOAP objects defined by the SOAP1.1 and SOAP 1.2 specifications. A `SOAPElement` object may be used to model the following:

- content in a `SOAPBody` object
- content in a `SOAPHeader` object
- content that can follow the `SOAPBody` object within a `SOAPEnvelope` object
- whatever may follow the detail element in a `SOAPFault` object

1.5 SOAPEnvelope & SOAPBody objects

The `SOAPEnvelope` object is a container object for the `SOAPHeader` and `SOAPBody` portions of a `SOAPPart` object. The `SOAPEnvelope` object must contain a `SOAPBody` object, but the `SOAPHeader` object is optional.

The `SOAPEnvelope` and `SOAPBody` objects both extend the `SOAPElement` object. The `SOAPBody` object models the contents of the SOAP body element in a SOAP message. A SOAP body element contains XML data that may determine how application-specific content must be processed.

1.6 SOAPBodyElement & SOAPFault

SOAPBody objects contain SOAPBodyElement objects that model the content of the SOAP body. An example of a SOAPBodyElement is the SOAPFault object.

1.7 SOAPFaultElement & Detail

The SOAPFaultElement is used to represent the contents of a SOAPFault object.

The Detail interface is a container for DetailEntry objects that provide application-specific error information associated with the SOAPBody object that contains it.

A Detail object is part of a SOAPFault object and may be retrieved using the getDetail method of the SOAPFault object.

The DetailEntry object extends SOAPElement and models the contents of a Detail object.

1.8 SOAPHeader & SOAPHeaderElement

A SOAPHeader object is an abstraction of the SOAP header element. A SOAPHeader object can be created using the SOAPEnvelope.addHeader method. SOAPHeader objects can have only SOAPHeaderElement objects as their immediate children. The addHeaderElement method creates a new HeaderElement object and adds it to the SOAPHeader object.

SOAPHeader and SOAPHeaderElement objects both extend the SOAPElement object. A SOAPHeaderElement object models the contents of the SOAP header of a SOAP envelope.

1.9 SOAPConnection & SOAPConnectionFactory

The `SOAPConnection` object represents a simple client-side view of a request-response style of SOAP messaging. A SAAJ client may choose to establish a synchronous point-to-point connection to a SOAP service using the `createConnection` method of the `SOAPConnectionFactory` object. Subsequently, a `SOAPMessage` may be sent to a remote party using the `call` method on the `SOAPConnection` object. Note that the `call` method will block until a `SOAPMessage` object is received.

A SAAJ based application may choose to use the `call` method to implement the client side of a simple point-to-point synchronous one-way message exchange scenario. In such a case, it is the application's responsibility to ignore the `SOAPMessage` object returned by the `call` method because the `SOAPMessage` object's only purpose is to unblock the client. It is assumed that a one-way service will not return a response to a request using the same connection when the `SOAPConnection.call` method was used to send the request.

SAAJ also provides support for the SOAP 1.2 Response Message Exchange Pattern (<http://www.w3.org/TR/2003/REC-soap12-part2-20030624/#soapresmep>) via the `SOAPConnection.get` method. This method can be used for pure information retrieval, where the representation of an available resource, identified by a URI, is fetched using a HTTP GET request without affecting the resource in any way

1.10 SOAPException object

The `SOAPException` object extends `java.lang.Exception` and is used to signal SOAP level exceptions.

1.11 Node & Text objects

The `Node` object models a node (element) of a DOM abstraction of an XML document.

The `Text` object extends `Node` and represents a node whose value is text. A `Text` object may model either text that is content or text that is a comment.

1.12 Name

The `Name` object models an XML name. This interface provides methods for getting the local names, namespace-qualified names, the prefix associated with the namespace for the name, and the URI of the namespace.

Name objects are created using the `SOAPEnvelope.createName` method.

1.13 SOAPFactory & SOAPElementFactory

These factories are intended primarily for the use of application components or tools that require the capability of inserting XML fragments into a SOAP Message. In SAAJ v1.1, the `SOAPElementFactory` has been deprecated in favor of `SOAPFactory` which serves as a super factory for the creation of `SOAPElement`, `Name`, and `Detail` objects.

1.14 SAAJMetaFactory

This Factory is the access point for the implementation classes of all the other factories defined in the SAAJ API. All of the `newInstance` methods defined on factories in SAAJ defer to instances of this class to do the actual object creation. The implementations of `newInstance()` methods (in `SOAPFactory` and `MessageFactory`) that existed in SAAJ 1.2 have been updated to also delegate to the `SAAJMetaFactory` when the SAAJ 1.2 defined lookup fails to locate the Factory implementation class name.

`SAAJMetaFactory` is a service provider interface. There are no public methods on this class.

1.15 SAAJResult

This concrete class acts as a holder for the results of a JAXP transformation or a JAXB marshalling, in the form of a SAAJ tree. This class will make it easier for the end user when dealing with transformations in situations where the result is expected to be a valid SAAJ tree. The results can be accessed by using the `getResult` method.

Package: javax.xml.soap

2.1 Description

Provides the API for creating and building SOAP messages. This package is defined in the *SOAP with Attachments API for Java™ (SAAJ) 1.4* specification.

The API in the `javax.xml.soap` package allows you to do the following:

- create a point-to-point connection to a specified endpoint
- create a SOAP message
- create an XML fragment
- add content to the header of a SOAP message
- add content to the body of a SOAP message
- create attachment parts and add content to them
- access/add/modify parts of a SOAP message
- create/add/modify SOAP fault information
- extract content from a SOAP message
- send a SOAP request-response message

In addition the APIs in the `javax.xml.soap` package extend their counterparts in the `org.w3c.dom` package. This means that the `SOAPPart` of a `SOAPMessage` is also a DOM Level 2 Document, and can be manipulated as such by applications, tools and libraries that use DOM (see <http://www.w3.org/DOM/> for more information). It is important to note that, while it is possible to use DOM APIs to add ordinary DOM nodes to a SAAJ tree, the SAAJ APIs are still required to return SAAJ types when examining or manipulating the tree. In order to accomplish this the SAAJ APIs (specifically `SOAPElement.getChildElements()` ⁷⁵) are allowed to silently replace objects that are

incorrectly typed relative to SAAJ requirements with equivalent objects of the required type. These replacements must never cause the logical structure of the tree to change, so from the perspective of the DOM APIs the tree will remain unchanged. However, the physical composition of the tree will have changed so that references to the nodes that were replaced will refer to nodes that are no longer a part of the tree. The SAAJ APIs are not allowed to make these replacements if they are not required so the replacement objects will never subsequently be silently replaced by future calls to the SAAJ API.

What this means in practical terms is that an application that starts to use SAAJ APIs on a tree after manipulating it using DOM APIs must assume that the tree has been translated into an all SAAJ tree and that any references to objects within the tree that were obtained using DOM APIs are no longer valid. Switching from SAAJ APIs to DOM APIs is not allowed to cause invalid references and neither is using SAAJ APIs exclusively. It is only switching from using DOM APIs on a particular SAAJ tree to using SAAJ APIs that causes the risk of invalid references.

Discovery of SAAJ implementation

There are several factories defined in the SAAJ API to discover and load specific implementation:

- SOAPFactory
- MessageFactory
- SOAPConnectionFactory
- SAAJMetaFactory

First three define newInstance() method which uses a common lookup procedure to determine the implementation class:

- Checks if a system property with the same name as the factory class is set (e.g. `javax.xml.soap.SOAPFactory`). If such property exists then its value is assumed to be the fully qualified name of the implementation class. This phase of the look up enables per-JVM override of the SAAJ implementation.
- Use the configuration file "jaxm.properties". The file is in standard `Properties` format and typically located in the `conf` directory of the Java installation. It contains the fully qualified name of the implementation class with the key being the system property defined above.
- Use the service-provider loading facilities, defined by the `ServiceLoader` class, to attempt to locate and load an implementation of the service using the default loading mechanism.
- Finally, if all the steps above fail, `SAAJMetaFactory` instance is used to locate specific implementation (for `MessageFactory` and `SOAPFactory`) or platform default implementation is used (`SOAPConnectionFactory`). Whenever `SAAJMetaFactory` is used, its lookup procedure to get actual instance is performed.

Class Summary

Interfaces

Detail ₁₇	A container for <code>DetailEntry</code> objects.
DetailEntry ₂₀	The content for a <code>Detail</code> object, giving details for a <code>SOAPFault</code> object.
Name ₃₂	A representation of an XML name.
Node ₃₅	A representation of a node (element) in an XML document.
SOAPBody ₄₆	An object that represents the contents of the SOAP body element in a SOAP message.
SOAPBodyElement ₅₄	A <code>SOAPBodyElement</code> object represents the contents in a <code>SOAPBody</code> object.
SOAPConstants ₆₁	The definition of constants pertaining to the SOAP 1.1 protocol.
SOAPElement ₆₇	An object representing an element of a SOAP message that is allowed but not specifically prescribed by a SOAP specification.
SOAPEnvelope ₈₄	The container for the <code>SOAPHeader</code> and <code>SOAPBody</code> portions of a <code>SOAPPart</code> object.
SOAPFault ₁₀₀	An element in the <code>SOAPBody</code> object that contains error and/or status information.
SOAPFaultElement ₁₁₃	A representation of the contents in a <code>SOAPFault</code> object.
SOAPHeader ₁₁₅	A representation of the SOAP header element.
SOAPHeaderElement ₁₂₃	An object representing the contents in the SOAP header part of the SOAP envelope.
Text ₁₄₆	A representation of a node whose value is text.

Classes

AttachmentPart ₅	A single attachment to a <code>SOAPMessage</code> object.
MessageFactory ₂₂	A factory for creating <code>SOAPMessage</code> objects.
MimeHeader ₂₆	An object that stores a MIME header name and its value.
MimeHeaders ₂₈	A container for <code>MimeHeader</code> objects, which represent the MIME headers present in a MIME part of a message.
SAAJMetaFactory ₃₉	The access point for the implementation classes of the factories defined in the SAAJ API.
SAAJResult ₄₂	Acts as a holder for the results of a JAXP transformation or a JAXB marshalling, in the form of a SAAJ tree.
SOAPConnection ₅₆	A point-to-point connection that a client can use for sending messages directly to a remote party (represented by a URL, for instance).
SOAPConnectionFactory ₅₉	A factory for creating <code>SOAPConnection</code> objects.
SOAPElementFactory ₈₁	<code>SOAPElementFactory</code> is a factory for XML fragments that will eventually end up in the SOAP part.
SOAPFactory ₉₃	<code>SOAPFactory</code> is a factory for creating various objects that exist in the SOAP XML tree.

Class Summary	
SOAPMessage₁₂₈	The root class for all SOAP messages.
SOAPPart₁₃₉	The container for the SOAP-specific portion of a SOAPMessage object.
Exceptions	
SOAPException₈₉	An exception that signals that a SOAP exception has occurred.

2.2 AttachmentPart

Declaration

```
public abstract class AttachmentPart

java.lang.Object
|
+-- javax.xml.soap.AttachmentPart
```

Description

A single attachment to a `SOAPMessage` object. A `SOAPMessage` object may contain zero, one, or many `AttachmentPart` objects. Each `AttachmentPart` object consists of two parts, application-specific content and associated MIME headers. The MIME headers consists of name/value pairs that can be used to identify and describe the content.

An `AttachmentPart` object must conform to certain standards.

1. It must conform to MIME [RFC2045] standards (<http://www.ietf.org/rfc/rfc2045.txt>)
2. It MUST contain content
3. The header portion MUST include the following header:
 - `Content-Type`
This header identifies the type of data in the content of an `AttachmentPart` object and MUST conform to [RFC2045]. The following is an example of a `Content-Type` header:

```
Content-Type: application/xml
```

The following line of code, in which `ap` is an `AttachmentPart` object, sets the header shown in the previous example.

```
ap.setMimeHeader("Content-Type", "application/xml");
```

There are no restrictions on the content portion of an `AttachmentPart` object. The content may be anything from a simple plain text object to a complex XML document or image file.

An `AttachmentPart` object is created with the method `SOAPMessage.createAttachmentPart`. After setting its MIME headers, the `AttachmentPart` object is added to the message that created it with the method `SOAPMessage.addAttachmentPart`.

The following code fragment, in which `m` is a `SOAPMessage` object and `contentString1` is a `String`, creates an instance of `AttachmentPart`, sets the `AttachmentPart` object with some content and header information, and adds the `AttachmentPart` object to the `SOAPMessage` object.

```
AttachmentPart ap1 = m.createAttachmentPart();
ap1.setContent(contentString1, "text/plain");
m.addAttachmentPart(ap1);
```

The following code fragment creates and adds a second `AttachmentPart` instance to the same message. `jpegData` is a binary byte buffer representing the `jpeg` file.

```
AttachmentPart ap2 = m.createAttachmentPart();
byte[] jpegData = ...;
ap2.setContent(new ByteArrayInputStream(jpegData), "image/jpeg");
m.addAttachmentPart(ap2);
```

The `getContent` method retrieves the contents and header from an `AttachmentPart` object. Depending on the `DataContentHandler` objects present, the returned `Object` can either be a typed Java object corresponding to the MIME type or an `InputStream` object that contains the content as bytes.

```
String content1 = ap1.getContent();
java.io.InputStream content2 = ap2.getContent();
```

The method `clearContent` removes all the content from an `AttachmentPart` object but does not affect its header information.

```
ap1.clearContent();
```

Member Summary	
Constructors	
	<code>AttachmentPart()</code> _g
Methods	
abstract void	<code>addMimeHeader(java.lang.String name, java.lang.String value)</code> _g Adds a MIME header with the specified name and value to this <code>AttachmentPart</code> object.
abstract void	<code>clearContent()</code> _g Clears out the content of this <code>AttachmentPart</code> object.
abstract	<code>getAllMimeHeaders()</code> _g Retrieves all the headers for this <code>AttachmentPart</code> object as an iterator over the <code>MimeHeader</code> objects.
java.util.Iterator<MimeHeader>	
abstract	<code>getBase64Content()</code> _g Returns an <code>InputStream</code> which can be used to obtain the content of <code>AttachmentPart</code> as Base64 encoded character data, this method would base64 encode the raw bytes of the attachment and return.
java.io.InputStream	
abstract	<code>getContent()</code> _g Gets the content of this <code>AttachmentPart</code> object as a Java object.
java.lang.Object	

Member Summary

java.lang.String	getContentId() ₁₀	Gets the value of the MIME header whose name is “Content-Id”.
java.lang.String	getContentLocation() ₁₀	Gets the value of the MIME header whose name is “Content-Location”.
java.lang.String	getContentType() ₁₀	Gets the value of the MIME header whose name is “Content-Type”.
abstract DataHandler	getDataHandler() ₁₀	Gets the DataHandler object for this AttachmentPart object.
abstract java.util.Iterator<MimeHeader>	getMatchingMimeHeaders(java.lang.String[] names) ₁₁	Retrieves all MimeHeader objects that match a name in the given array.
abstract java.lang.String[]	getMimeHeader(java.lang.String name) ₁₁	Gets all the values of the header identified by the given String.
abstract java.util.Iterator<MimeHeader>	getNonMatchingMimeHeaders(java.lang.String[] names) ₁₁	Retrieves all MimeHeader objects whose name does not match a name in the given array.
abstract java.io.InputStream	getRawContent() ₁₂	Gets the content of this AttachmentPart object as an InputStream as if a call had been made to getContent and no DataContentHandler had been registered for the content-type of this AttachmentPart.
abstract byte[]	getRawContentBytes() ₁₂	Gets the content of this AttachmentPart object as a byte[] array as if a call had been made to getContent and no DataContentHandler had been registered for the content-type of this AttachmentPart.
abstract int	getSize() ₁₃	Returns the number of bytes in this AttachmentPart object.
abstract void	removeAllMimeHeaders() ₁₃	Removes all the MIME header entries.
abstract void	removeMimeHeader(java.lang.String header) ₁₃	Removes all MIME headers that match the given name.
abstract void	setBase64Content(java.io.InputStream content, java.lang.String contentType) ₁₃	Sets the content of this attachment part from the Base64 source InputStream and sets the value of the Content-Type header to the value contained in contentType. This method would first decode the base64 input and write the resulting raw bytes to the attachment.
abstract void	setContent(java.lang.Object object, java.lang.String contentType) ₁₄	Sets the content of this attachment part to that of the given Object and sets the value of the Content-Type header to the given type.
void	setContentId(java.lang.String contentId) ₁₄	Sets the MIME header whose name is “Content-Id” with the given value.
void	setContentLocation(java.lang.String contentLocation) ₁₄	Sets the MIME header whose name is “Content-Location” with the given value.
void	setContentType(java.lang.String contentType) ₁₅	Sets the MIME header whose name is “Content-Type” with the given value.

Member Summary	
abstract void	<code>setDataHandler(DataHandler dataHandler)</code> ¹⁵ Sets the given DataHandler object as the data handler for this AttachmentPart object.
abstract void	<code>setMimeHeader(java.lang.String name, java.lang.String value)</code> ¹⁵ Changes the first header entry that matches the given name to the given value, adding a new header if no existing header matches.
abstract void	<code>setRawContent(java.io.InputStream content, java.lang.String contentType)</code> ¹⁶ Sets the content of this attachment part to that contained by the InputStream content and sets the value of the Content-Type header to the value contained in contentType.
abstract void	<code>setRawContentBytes(byte[] content, int offset, int len, java.lang.String contentType)</code> ¹⁶ Sets the content of this attachment part to that contained by the byte [] array content and sets the value of the Content-Type header to the value contained in contentType.

Inherited Member Summary
Methods inherited from class Object <code>clone()</code> , <code>equals(Object)</code> , <code>finalize()</code> , <code>getClass()</code> , <code>hashCode()</code> , <code>notify()</code> , <code>notifyAll()</code> , <code>toString()</code> , <code>wait(long, int)</code> , <code>wait(long, int)</code> , <code>wait(long, int)</code>

Constructors

AttachmentPart()

```
public AttachmentPart()
```

Methods

addMimeHeader(String, String)

```
public abstract void addMimeHeader(java.lang.String name, java.lang.String value)
```

Adds a MIME header with the specified name and value to this AttachmentPart object.

Note that RFC822 headers can contain only US-ASCII characters.

Parameters:

name - a String giving the name of the header to be added

value - a String giving the value of the header to be added

Throws:

`java.lang.IllegalArgumentException` - if there was a problem with the specified mime header name or value

clearContent()

```
public abstract void clearContent()
```

Clears out the content of this `AttachmentPart` object. The MIME header portion is left untouched.

getAllMimeHeaders()

```
public abstract java.util.Iterator<MimeHeader> getAllMimeHeaders()
```

Retrieves all the headers for this `AttachmentPart` object as an iterator over the `MimeHeader` objects.

Returns: an `Iterator` object with all of the Mime headers for this `AttachmentPart` object

getBase64Content()

```
public abstract java.io.InputStream getBase64Content()  
    throws SOAPException
```

Returns an `InputStream` which can be used to obtain the content of `AttachmentPart` as Base64 encoded character data, this method would base64 encode the raw bytes of the attachment and return.

Returns: an `InputStream` from which the Base64 encoded `AttachmentPart` can be read.

Throws:

`SOAPException`₈₉ - if there is no content set into this `AttachmentPart` object or if there was a data transformation error.

Since: SAAJ 1.3

getContent()

```
public abstract java.lang.Object getContent()  
    throws SOAPException
```

Gets the content of this `AttachmentPart` object as a Java object. The type of the returned Java object depends on (1) the `DataContentHandler` object that is used to interpret the bytes and (2) the `Content-Type` given in the header.

For the MIME content types “text/plain”, “text/html” and “text/xml”, the `DataContentHandler` object does the conversions to and from the Java types corresponding to the MIME types. For other MIME types, the `DataContentHandler` object can return an `InputStream` object that contains the content data as raw bytes.

A SAAJ-compliant implementation must, as a minimum, return a `java.lang.String` object corresponding to any content stream with a `Content-Type` value of `text/plain`, a `javax.xml.transform.stream.StreamSource` object corresponding to a content stream with a `Content-Type` value of `text/xml`, a `java.awt.Image` object corresponding to a content stream with a `Content-Type` value of `image/gif` or `image/jpeg`. For those content types that an installed `DataContentHandler` object does not understand, the `DataContentHandler` object is required to return a `java.io.InputStream` object with the raw bytes.

Returns: a Java object with the content of this `AttachmentPart` object

Throws:

`SOAPException`₈₉ - if there is no content set into this `AttachmentPart` object or if there was a data transformation error

getContentId()

```
public java.lang.String getContentId()
```

Gets the value of the MIME header whose name is “Content-Id”.

Returns: a `String` giving the value of the “Content-Id” header or `null` if there is none

See Also: `setContentId(String)`₁₄

getContentLocation()

```
public java.lang.String getContentLocation()
```

Gets the value of the MIME header whose name is “Content-Location”.

Returns: a `String` giving the value of the “Content-Location” header or `null` if there is none

getContentType()

```
public java.lang.String getContentType()
```

Gets the value of the MIME header whose name is “Content-Type”.

Returns: a `String` giving the value of the “Content-Type” header or `null` if there is none

getDataHandler()

```
public abstract DataHandler getDataHandler()  
    throws SOAPException
```

Gets the `DataHandler` object for this `AttachmentPart` object.

Returns: the `DataHandler` object associated with this `AttachmentPart` object

Throws:

`SOAPException99` - if there is no data in this `AttachmentPart` object

getMatchingMimeHeaders(String[])

```
public abstract java.util.Iterator<MimeHeader>  
    getMatchingMimeHeaders (java.lang.String[] names)
```

Retrieves all `MimeHeader` objects that match a name in the given array.

Parameters:

names - a `String` array with the name(s) of the MIME headers to be returned

Returns: all of the MIME headers that match one of the names in the given array as an `Iterator` object

getMimeHeader(String)

```
public abstract java.lang.String[] getMimeHeader (java.lang.String name)
```

Gets all the values of the header identified by the given `String`.

Parameters:

name - the name of the header; example: "Content-Type"

Returns: a `String` array giving the value for the specified header

See Also: `setMimeHeader(String, String)15`

getNonMatchingMimeHeaders(String[])

```
public abstract java.util.Iterator<MimeHeader>  
    getNonMatchingMimeHeaders (java.lang.String[] names)
```

Retrieves all `MimeHeader` objects whose name does not match a name in the given array.

Parameters:

names - a `String` array with the name(s) of the MIME headers not to be returned

Returns: all of the MIME headers in this `AttachmentPart` object except those that match one of the names in the given array. The nonmatching MIME headers are returned as an `Iterator` object.

getRawContent()

```
public abstract java.io.InputStream getRawContent()  
    throws SOAPException
```

Gets the content of this `AttachmentPart` object as an `InputStream` as if a call had been made to `getContent` and no `DataContentHandler` had been registered for the `content-type` of this `AttachmentPart`.

Note that reading from the returned `InputStream` would result in consuming the data in the stream. It is the responsibility of the caller to reset the `InputStream` appropriately before calling a Subsequent API. If a copy of the raw attachment content is required then the `getRawContentBytes()`¹² API should be used instead.

Returns: an `InputStream` from which the raw data contained by the `AttachmentPart` can be accessed.

Throws:

`SOAPException`⁸⁹ - if there is no content set into this `AttachmentPart` object or if there was a data transformation error.

Since: SAAJ 1.3

See Also: `getRawContentBytes()`¹²

getRawContentBytes()

```
public abstract byte[] getRawContentBytes()  
    throws SOAPException
```

Gets the content of this `AttachmentPart` object as a `byte[]` array as if a call had been made to `getContent` and no `DataContentHandler` had been registered for the `content-type` of this `AttachmentPart`.

Returns: a `byte []` array containing the raw data of the `AttachmentPart`.

Throws:

`SOAPException`⁸⁹ - if there is no content set into this `AttachmentPart` object or if there was a data transformation error.

Since: SAAJ 1.3

getSize()

```
public abstract int getSize()  
    throws SOAPException
```

Returns the number of bytes in this AttachmentPart object.

Returns: the size of this AttachmentPart object in bytes or -1 if the size cannot be determined

Throws:

[SOAPException](#)₈₉ - if the content of this attachment is corrupted or if there was an exception while trying to determine the size.

removeAllMimeHeaders()

```
public abstract void removeAllMimeHeaders()
```

Removes all the MIME header entries.

removeMimeHeader(String)

```
public abstract void removeMimeHeader(java.lang.String header)
```

Removes all MIME headers that match the given name.

Parameters:

header - the string name of the MIME header/s to be removed

setBase64Content(Reader, String)

```
public abstract void setBase64Content(java.io.InputStream content,  
    java.lang.String contentType) throws SOAPException
```

Sets the content of this attachment part from the Base64 source InputStream and sets the value of the Content-Type header to the value contained in contentType, This method would first decode the base64 input and write the resulting raw bytes to the attachment.

A subsequent call to [getSize\(\)](#)₁₃ may not be an exact measure of the content size.

Parameters:

content - the base64 encoded data to add to the attachment part

contentType - the value to set into the Content-Type header

Throws:

SOAPException - if there is an error in setting the content

java.lang.NullPointerException - if content is null

Since: SAAJ 1.3

setContent(Object, String)

```
public abstract void setContent(java.lang.Object object,  
                                java.lang.String contentType)
```

Sets the content of this attachment part to that of the given `Object` and sets the value of the Content-Type header to the given type. The type of the `Object` should correspond to the value given for the Content-Type. This depends on the particular set of `DataContentHandler` objects in use.

Parameters:

`object` - the Java object that makes up the content for this attachment part

`contentType` - the MIME string that specifies the type of the content

Throws:

`java.lang.IllegalArgumentException` - may be thrown if the `contentType` does not match the type of the content object, or if there was no `DataContentHandler` object for this content object

See Also: [getContent\(\)](#)₉

setContentId(String)

```
public void setContentId(java.lang.String contentId)
```

Sets the MIME header whose name is “Content-Id” with the given value.

Parameters:

`contentId` - a `String` giving the value of the “Content-Id” header

Throws:

`java.lang.IllegalArgumentException` - if there was a problem with the specified `contentId` value

See Also: [getContentId\(\)](#)₁₀

setContentLocation(String)

```
public void setContentLocation(java.lang.String contentLocation)
```

Sets the MIME header whose name is “Content-Location” with the given value.

Parameters:

`contentLocation` - a `String` giving the value of the “Content-Location” header

Throws:

`java.lang.IllegalArgumentException` - if there was a problem with the specified content location

setContentType(String)

```
public void setContentType(java.lang.String contentType)
```

Sets the MIME header whose name is “Content-Type” with the given value.

Parameters:

contentType - a String giving the value of the “Content-Type” header

Throws:

java.lang.IllegalArgumentException - if there was a problem with the specified content type

setDataHandler(DataHandler)

```
public abstract void setDataHandler(DataHandler dataHandler)
```

Sets the given DataHandler object as the data handler for this AttachmentPart object. Typically, on an incoming message, the data handler is automatically set. When a message is being created and populated with content, the setDataHandler method can be used to get data from various data sources into the message.

Parameters:

dataHandler - the DataHandler object to be set

Throws:

java.lang.IllegalArgumentException - if there was a problem with the specified DataHandler object

setMimeHeader(String, String)

```
public abstract void setMimeHeader(java.lang.String name, java.lang.String value)
```

Changes the first header entry that matches the given name to the given value, adding a new header if no existing header matches. This method also removes all matching headers but the first.

Note that RFC822 headers can only contain US-ASCII characters.

Parameters:

name - a String giving the name of the header for which to search

value - a String giving the value to be set for the header whose name matches the given name

Throws:

java.lang.IllegalArgumentException - if there was a problem with the specified mime header name or value

setRawContent(InputStream, String)

```
public abstract void setRawContent(java.io.InputStream content,  
    java.lang.String contentType) throws SOAPException
```

Sets the content of this attachment part to that contained by the `InputStream` `content` and sets the value of the Content-Type header to the value contained in `contentType`.

A subsequent call to `getSize()`₁₃ may not be an exact measure of the content size.

Parameters:

`content` - the raw data to add to the attachment part

`contentType` - the value to set into the Content-Type header

Throws:

`SOAPException` - if there is an error in setting the content

`java.lang.NullPointerException` - if `content` is null

Since: SAAJ 1.3

setRawContentBytes(byte[], int, int, String)

```
public abstract void setRawContentBytes(byte[] content, int offset, int len,  
    java.lang.String contentType)  
    throws SOAPException
```

Sets the content of this attachment part to that contained by the `byte[]` array `content` and sets the value of the Content-Type header to the value contained in `contentType`.

Parameters:

`content` - the raw data to add to the attachment part

`contentType` - the value to set into the Content-Type header

`offset` - the offset in the byte array of the content

`len` - the number of bytes that form the content

Throws:

`SOAPException`₈₉ - if an there is an error in setting the content or content is null

Since: SAAJ 1.3

2.3 Detail

Declaration

```
public interface Detail extends SOAPFaultElement113
```

All Superinterfaces: org.w3c.dom.Element, org.w3c.dom.Node, Node₃₅,
SOAPElement₆₇, SOAPFaultElement₁₁₃

Description

A container for DetailEntry objects. DetailEntry objects give detailed error information that is application-specific and related to the SOAPBody object that contains it.

A Detail object, which is part of a SOAPFault object, can be retrieved using the method SOAPFault.getDetail. The Detail interface provides two methods. One creates a new DetailEntry object and also automatically adds it to the Detail object. The second method gets a list of the DetailEntry objects contained in a Detail object.

The following code fragment, in which *sf* is a SOAPFault object, gets its Detail object (*d*), adds a new DetailEntry object to *d*, and then gets a list of all the DetailEntry objects in *d*. The code also creates a Name object to pass to the method addDetailEntry. The variable *se*, used to create the Name object, is a SOAPEnvelope object.

```
Detail d = sf.getDetail();
Name name = se.createName("GetLastTradePrice", "WOMBAT",
                          "http://www.wombat.org/trader");

d.addDetailEntry(name);
Iterator<DetailEntry> it = d.getDetailEntries();
```

Member Summary

Methods

- | | |
|-------------|---|
| DetailEntry | <code>addDetailEntry(Name name)₁₉</code>
Creates a new DetailEntry object with the given name and adds it to this Detail object. |
| DetailEntry | <code>addDetailEntry(javax.xml.namespace.QName qname)₁₉</code>
Creates a new DetailEntry object with the given QName and adds it to this Detail object. |

Member Summary

`java.util.Iterator<DetailEntry>` [getDetailEntries\(\)](#)₁₉ Gets an Iterator over all of the DetailEntries in this Detail object.

Inherited Member Summary

Fields inherited from interface `Node`

`ATTRIBUTE_NODE`, `CDATA_SECTION_NODE`, `COMMENT_NODE`, `DOCUMENT_FRAGMENT_NODE`, `DOCUMENT_NODE`, `DOCUMENT_TYPE_NODE`, `ELEMENT_NODE`, `ENTITY_NODE`, `ENTITY_REFERENCE_NODE`, `NOTATION_NODE`, `PROCESSING_INSTRUCTION_NODE`, `TEXT_NODE`

Methods inherited from interface `Element`

`getAttribute(String)`, `getAttributeNS(String, String)`, `getAttributeNode(String)`, `getAttributeNodeNS(String, String)`, `getElementsByName(String)`, `getElementsByTagName(String)`, `getTagName()`, `hasAttribute(String)`, `hasAttributeNS(String, String)`, `removeAttribute(String)`, `removeAttributeNS(String, String)`, `removeAttributeNode(Attr)`, `setAttribute(String, String)`, `setAttributeNS(String, String, String)`, `setAttributeNode(Attr)`, `setAttributeNodeNS(Attr)`

Methods inherited from interface `Node`₃₅

[detachNode\(\)](#)₃₆, [getParentElement\(\)](#)₃₆, [getValue\(\)](#)₃₆, [recycleNode\(\)](#)₃₇, [setParentElement\(SOAPElement\)](#)₃₇, [setValue\(String\)](#)₃₇

Methods inherited from interface `Node`

`appendChild(Node)`, `cloneNode(boolean)`, `getAttributes()`, `getChildNodes()`, `getFirstChild()`, `getLastChild()`, `getLocalName()`, `getNamespaceURI()`, `getNextSibling()`, `getNodeName()`, `getNodeType()`, `getNodeValue()`, `getOwnerDocument()`, `getParentNode()`, `getPrefix()`, `getPreviousSibling()`, `hasAttributes()`, `hasChildNodes()`, `insertBefore(Node, Node)`, `isSupported(String, String)`, `normalize()`, `removeChild(Node)`, `replaceChild(Node, Node)`, `setNodeValue(String)`, `setPrefix(String)`

Methods inherited from interface `SOAPElement`₆₇

[addAttribute\(Name, String\)](#)₇₀, [addChildElement\(SOAPElement\)](#)₇₂, [addChildElement\(SOAPElement\)](#)₇₂, [addChildElement\(SOAPElement\)](#)₇₂, [addChildElement\(SOAPElement\)](#)₇₂, [addNamespaceDeclaration\(String, String\)](#)₇₃, [addTextNode\(String\)](#)₇₄, [getAllAttributes\(\)](#)₇₄, [getAttributeValue\(Name\)](#)₇₅, [getChildElements\(Name\)](#)₇₆, [getChildElements\(Name\)](#)₇₆, [getElementName\(\)](#)₇₇, [getEncodingStyle\(\)](#)₇₇, [getNamespacePrefixes\(\)](#)₇₇, [getNamespaceURI\(String\)](#)₇₈, [getVisibleNamespacePrefixes\(\)](#)₇₈, [removeAttribute\(Name\)](#)₇₈, [removeContents\(\)](#)₇₉, [removeNamespaceDeclaration\(String\)](#)₇₉, [setEncodingStyle\(String\)](#)₇₉

Methods

addDetailEntry(Name)

```
public javax.xml.soap.DetailEntry20 addDetailEntry(javax.xml.soap.Name32 name)
    throws SOAPException
```

Creates a new `DetailEntry` object with the given name and adds it to this `Detail` object.

Parameters:

name - a `Name` object identifying the new `DetailEntry` object

Returns: the new `DetailEntry` object that was created

Throws:

`SOAPException`₈₉ - thrown when there is a problem in adding a `DetailEntry` object to this `Detail` object.

See Also: [addDetailEntry\(QName\)](#)₁₉

addDetailEntry(QName)

```
public javax.xml.soap.DetailEntry20 addDetailEntry(javax.xml.namespace.QName qname)
    throws SOAPException
```

Creates a new `DetailEntry` object with the given `QName` and adds it to this `Detail` object. This method is the preferred over the one using `Name`.

Parameters:

qname - a `QName` object identifying the new `DetailEntry` object

Returns: the new `DetailEntry` object that was created

Throws:

`SOAPException`₈₉ - thrown when there is a problem in adding a `DetailEntry` object to this `Detail` object.

Since: SAAJ 1.3

See Also: [addDetailEntry\(Name\)](#)₁₉

getDetailEntries()

```
public java.util.Iterator<DetailEntry> getDetailEntries()
```

Gets an `Iterator` over all of the `DetailEntry`s in this `Detail` object.

Returns: an `Iterator` object over the `DetailEntry` objects in this `Detail` object

2.4 DetailEntry

Declaration

public interface **DetailEntry** extends [SOAPElement](#)₆₇

All Superinterfaces: [org.w3c.dom.Element](#), [org.w3c.dom.Node](#), [Node](#)₃₅,
[SOAPElement](#)₆₇

Description

The content for a Detail object, giving details for a SOAPFault object. A DetailEntry object, which carries information about errors related to the SOAPBody object that contains it, is application-specific.

Inherited Member Summary
<p>Fields inherited from interface Node</p> <p>ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE</p> <p>Methods inherited from interface Element</p> <p>getAttribute(String), getAttributeNS(String, String), getAttributeNode(String), getAttributeNodeNS(String, String), getElementsByTagName(String), getElementsByTagNameNS(String, String), getTagName(), hasAttribute(String), hasAttributeNS(String, String), removeAttribute(String), removeAttributeNS(String, String), removeAttributeNode(Attr), setAttribute(String, String), setAttributeNS(String, String, String), setAttributeNode(Attr), setAttributeNodeNS(Attr)</p> <p>Methods inherited from interface Node₃₅</p> <p>detachNode()₃₆, getParentElement()₃₆, getValue()₃₆, recycleNode()₃₇, setParentElement(SOAPElement)₃₇, setValue(String)₃₇</p> <p>Methods inherited from interface Node</p>

Inherited Member Summary

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(),
getFirstChild(), getLastChild(), getLocalName(), getNamespaceURI(), getNextSibling(),
getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(),
getPrefix(), getPreviousSibling(), hasAttributes(), hasChildNodes(),
insertBefore(Node, Node), isSupported(String, String), normalize(),
removeChild(Node), replaceChild(Node, Node), setNodeValue(String), setPrefix(String)

Methods inherited from interface [SOAPElement](#)₆₇

addAttribute(Name, String)₇₀, addChildElement(SOAPElement)₇₂,
addChildElement(SOAPElement)₇₂, addChildElement(SOAPElement)₇₂,
addChildElement(SOAPElement)₇₂, addChildElement(SOAPElement)₇₂,
addNamespaceDeclaration(String, String)₇₃, addTextNode(String)₇₄,
getAllAttributes()₇₄, getAttributeValue(Name)₇₅, getChildElements(Name)₇₆,
getChildElements(Name)₇₆, getElementName()₇₇, getEncodingStyle()₇₇,
getNamespacePrefixes()₇₇, getNamespaceURI(String)₇₈, getVisibleNamespacePrefixes()₇₈,
removeAttribute(Name)₇₈, removeContents()₇₉, removeNamespaceDeclaration(String)₇₉,
setEncodingStyle(String)₇₉

2.5 MessageFactory

Declaration

```
public abstract class MessageFactory
```

```
java.lang.Object  
|  
+--javax.xml.soap.MessageFactory
```

Description

A factory for creating SOAPMessage objects.

A SAAJ client can create a MessageFactory object using the method newInstance, as shown in the following lines of code.

```
MessageFactory mf = MessageFactory.newInstance();  
MessageFactory mf12 = MessageFactory.newInstance(SOAPConstants.SOAP_1_2_PROTOCOL);
```

All MessageFactory objects, regardless of how they are created, will produce SOAPMessage objects that have the following elements by default:

- A SOAPPart object
- A SOAPEnvelope object
- A SOAPBody object
- A SOAPHeader object

In some cases, specialized MessageFactory objects may be obtained that produce messages prepopulated with additional entries in the SOAPHeader object and the SOAPBody object. The content of a new SOAPMessage object depends on which of the two MessageFactory methods is used to create it.

- createMessage()
This is the method clients would normally use to create a request message.
- createMessage(MimeHeaders, java.io.InputStream) — message has content from the InputStream object and headers from the MimeHeaders object.
This method can be used internally by a service implementation to create a message that is a response to a request.

Member Summary	
Constructors	
	<code>MessageFactory()</code> ²³
Methods	
abstract SOAPMessage	<code>createMessage()</code> ²³ Creates a new SOAPMessage object with the default SOAPPart, SOAPEnvelope, SOAPBody, and SOAPHeader objects.
abstract SOAPMessage	<code>createMessage(MimeHeaders headers, java.io.InputStream in)</code> ²⁴ Internalizes the contents of the given InputStream object into a new SOAPMessage object and returns the SOAPMessage object.
static MessageFactory	<code>newInstance()</code> ²⁴ Creates a new MessageFactory object that is an instance of the default implementation (SOAP 1.1). This method uses the lookup procedure specified in javax.xml.soap to locate and load the MessageFactory class.
static MessageFactory	<code>newInstance(java.lang.String protocol)</code> ²⁵ Creates a new MessageFactory object that is an instance of the specified implementation.

Inherited Member Summary
Methods inherited from class Object
<code>clone(), equals(Object), finalize(), getClass(), hashCode(), notify(), notifyAll(), toString(), wait(long, int), wait(long, int), wait(long, int)</code>

Constructors

MessageFactory()

```
public MessageFactory()
```

Methods

createMessage()

```
public abstract javax.xml.soap.SOAPMessage 128 createMessage()
    throws SOAPException
```

Creates a new SOAPMessage object with the default SOAPPart, SOAPEnvelope, SOAPBody, and SOAPHeader objects. Profile-specific message factories can choose to prepopulate the SOAPMessage object with profile-specific headers.

Content can be added to this message's SOAPPart object, and the message can be sent “as is” when a message containing only a SOAP part is sufficient. Otherwise, the SOAPMessage object

needs to create one or more `AttachmentPart` objects and add them to itself. Any content that is not in XML format must be in an `AttachmentPart` object.

Returns: a new `SOAPMessage` object

Throws:

`SOAPException`₈₉ - if a SOAP error occurs

`java.lang.UnsupportedOperationException` - if the protocol of this `MessageFactory` instance is `DYNAMIC_SOAP_PROTOCOL`₆₂

createMessage(MimeHeaders, InputStream)

```
public abstract javax.xml.soap.SOAPOutputMessage128
    createMessage(javax.xml.soap.MimeHeaders28 headers,
        java.io.InputStream in)
    throws IOException, SOAPException
```

Internalizes the contents of the given `InputStream` object into a new `SOAPMessage` object and returns the `SOAPMessage` object.

Parameters:

`in` - the `InputStream` object that contains the data for a message

`headers` - the transport-specific headers passed to the message in a transport-independent fashion for creation of the message

Returns: a new `SOAPMessage` object containing the data from the given `InputStream` object

Throws:

`java.io.IOException` - if there is a problem in reading data from the input stream

`SOAPException`₈₉ - may be thrown if the message is invalid

`java.lang.IllegalArgumentException` - if the `MessageFactory` requires one or more MIME headers to be present in the `headers` parameter and they are missing.

`MessageFactory` implementations for `SOAP_1_1_PROTOCOL` or

`SOAP_1_2_PROTOCOL` must not throw `IllegalArgumentException` for this reason.

newInstance()

```
public static javax.xml.soap.MessageFactory22 newInstance()
    throws SOAPException
```

Creates a new `MessageFactory` object that is an instance of the default implementation (`SOAP 1.1`). This method uses the lookup procedure specified in `javax.xml.soap` to locate and load `MessageFactory` class.

Returns: a new instance of a `MessageFactory`

Throws:

[SOAPException₈₉](#) - if there was an error in creating the default implementation of the `MessageFactory`.

See Also: [SAAJMetaFactory₃₉](#)

newInstance(String)

```
public static javax.xml.soap.MessageFactory22 newInstance(java.lang.String
    protocol)
    throws SOAPException
```

Creates a new `MessageFactory` object that is an instance of the specified implementation. May be a dynamic message factory, a SOAP 1.1 message factory, or a SOAP 1.2 message factory. A dynamic message factory creates messages based on the MIME headers specified as arguments to the `createMessage` method. This method uses the [SAAJMetaFactory₃₉](#) to locate the implementation class and create the `MessageFactory` instance.

Parameters:

`protocol` - a string constant representing the class of the specified message factory implementation. May be either `DYNAMIC_SOAP_PROTOCOL`, `DEFAULT_SOAP_PROTOCOL` (which is the same as) `SOAP_1_1_PROTOCOL`, or `SOAP_1_2_PROTOCOL`.

Returns: a new instance of a `MessageFactory`

Throws:

[SOAPException₈₉](#) - if there was an error in creating the specified implementation of `MessageFactory`.

See Also: [SAAJMetaFactory₃₉](#)

Since: SAAJ 1.3

2.6 MimeHeader

Declaration

```
public class MimeHeader
    java.lang.Object
    |
    +--javax.xml.soap.MimeHeader
```

Description

An object that stores a MIME header name and its value. One or more `MimeHeader` objects may be contained in a `MimeHeaders` object.

See Also: [MimeHeaders₂₈](#)

Member Summary	
Constructors	
	<div><code>MimeHeader(java.lang.String name, java.lang.String value)₂₇</code> Constructs a <code>MimeHeader</code> object initialized with the given name and value.</div>
Methods	
java.lang.String	<div><code>getName()₂₇</code> Returns the name of this <code>MimeHeader</code> object.</div>
java.lang.String	<div><code>getValue()₂₇</code> Returns the value of this <code>MimeHeader</code> object.</div>

Inherited Member Summary
Methods inherited from class Object <code>clone(), equals(Object), finalize(), getClass(), hashCode(), notify(), notifyAll(), toString(), wait(long, int), wait(long, int), wait(long, int)</code>

Constructors

MimeHeader(String, String)

```
public MimeHeader(java.lang.String name, java.lang.String value)
```

Constructs a MimeHeader object initialized with the given name and value.

Parameters:

name - a String giving the name of the header

value - a String giving the value of the header

Methods

getName()

```
public java.lang.String getName()
```

Returns the name of this MimeHeader object.

Returns: the name of the header as a String

getValue()

```
public java.lang.String getValue()
```

Returns the value of this MimeHeader object.

Returns: the value of the header as a String

2.7 MimeHeaders

Declaration

```
public class MimeHeaders
    java.lang.Object
    |
    +-- javax.xml.soap.MimeHeaders
```

Description

A container for MimeHeader objects, which represent the MIME headers present in a MIME part of a message.

This class is used primarily when an application wants to retrieve specific attachments based on certain MIME headers and values. This class will most likely be used by implementations of AttachmentPart and other MIME dependent parts of the SAAJ API.

See Also: [SOAPMessage.getAttachments\(\)](#)₁₃₄, [AttachmentPart](#)₅

Member Summary		
Constructors		
	MimeHeaders()	₂₉ Constructs a default MimeHeaders object initialized with an empty Vector object.
Methods		
	void addHeader(java.lang.String name, java.lang.String value)	₂₉ Adds a MimeHeader object with the specified name and value to this MimeHeaders object's list of headers.
java.util.Iterator<MimeHeader>	getAllHeaders()	₃₀ Returns all the MimeHeaders in this MimeHeaders object.
java.lang.String[]	getHeader(java.lang.String name)	₃₀ Returns all of the values for the specified header as an array of String objects.
java.util.Iterator<MimeHeader>	getMatchingHeaders(java.lang.String names)	₃₀ Returns all the MimeHeader objects whose name matches a name in the given array of names.

Member Summary	
java.util.Iterator<MimeHeader>	<code>getNonMatchingHeaders(java.lang.String names)</code> ₃₀ Returns all of the MimeHeader objects whose name does not match a name in the given array of names.
void	<code>removeAllHeaders()</code> ₃₁ Removes all the header entries from this MimeHeaders object.
void	<code>removeHeader(java.lang.String name)</code> ₃₁ Remove all MimeHeader objects whose name matches the given name.
void	<code>setHeader(java.lang.String name, java.lang.String value)</code> ₃₁ Replaces the current value of the first header entry whose name matches the given name with the given value, adding a new header if no existing header name matches.

Inherited Member Summary
Methods inherited from class Object <code>clone(), equals(Object), finalize(), getClass(), hashCode(), notify(), notifyAll(), toString(), wait(long, int), wait(long, int), wait(long, int)</code>

Constructors

MimeHeaders()

```
public MimeHeaders()
```

Constructs a default MimeHeaders object initialized with an empty Vector object.

Methods

addHeader(String, String)

```
public void addHeader(java.lang.String name, java.lang.String value)
```

Adds a MimeHeader object with the specified name and value to this MimeHeaders object's list of headers.

Note that RFC822 headers can contain only US-ASCII characters.

Parameters:

name - a String with the name of the header to be added

value - a String with the value of the header to be added

Throws:

`java.lang.IllegalArgumentException` - if there was a problem in the mime header name or value being added

getAllHeaders()

```
public java.util.Iterator<MimeHeader> getAllHeaders()
```

Returns all the MimeHeaders in this MimeHeaders object.

Returns: an Iterator object over this MimeHeaders object's list of MimeHeader objects

getHeader(String)

```
public java.lang.String[] getHeader(java.lang.String name)
```

Returns all of the values for the specified header as an array of String objects.

Parameters:

name - the name of the header for which values will be returned

Returns: a String array with all of the values for the specified header

See Also: [setHeader\(String, String\)](#)₃₁

getMatchingHeaders(String[])

```
public java.util.Iterator<MimeHeader> getMatchingHeaders(java.lang.String[] names)
```

Returns all the MimeHeader objects whose name matches a name in the given array of names.

Parameters:

names - an array of String objects with the names for which to search

Returns: an Iterator object over the MimeHeader objects whose name matches one of the names in the given list

getNonMatchingHeaders(String[])

```
public java.util.Iterator<MimeHeader> getNonMatchingHeaders(java.lang.String[]  
names)
```

Returns all of the MimeHeader objects whose name does not match a name in the given array of names.

Parameters:

names - an array of String objects with the names for which to search

Returns: an Iterator object over the MimeHeader objects whose name does not match one of the names in the given list

removeAllHeaders()

```
public void removeAllHeaders()
```

Removes all the header entries from this `MimeHeaders` object.

removeHeader(String)

```
public void removeHeader(java.lang.String name)
```

Remove all `MimeHeader` objects whose name matches the given name.

Parameters:

name - a `String` with the name of the header for which to search

setHeader(String, String)

```
public void setHeader(java.lang.String name, java.lang.String value)
```

Replaces the current value of the first header entry whose name matches the given name with the given value, adding a new header if no existing header name matches. This method also removes all matching headers after the first one.

Note that RFC822 headers can contain only US-ASCII characters.

Parameters:

name - a `String` with the name of the header for which to search

value - a `String` with the value that will replace the current value of the specified header

Throws:

`java.lang.IllegalArgumentException` - if there was a problem in the mime header name or the value being set

See Also: [getHeader\(String\)](#)₃₀

2.8 Name

Declaration

```
public interface Name
```

Description

A representation of an XML name. This interface provides methods for getting the local and namespace-qualified names and also for getting the prefix associated with the namespace for the name. It is also possible to get the URI of the namespace.

The following is an example of a namespace declaration in an element.

```
<wombat:GetLastTradePrice xmlns:wombat="http://www.wombat.org/trader">
```

(“xmlns” stands for “XML namespace”) The following shows what the methods in the Name interface will return.

- `getQualifiedName` will return “prefix:LocalName” = “WOMBAT:GetLastTradePrice”
- `getURI` will return “http://www.wombat.org/trader”
- `getLocalName` will return “GetLastTracePrice”
- `getPrefix` will return “WOMBAT”

XML namespaces are used to disambiguate SOAP identifiers from application-specific identifiers.

Name objects are created using the method `SOAPEnvelope.createName`, which has two versions. One method creates Name objects with a local name, a namespace prefix, and a namespace URI. and the second creates Name objects with just a local name. The following line of code, in which *se* is a `SOAPEnvelope` object, creates a new Name object with all three.

```
Name name = se.createName("GetLastTradePrice", "WOMBAT",  
                           "http://www.wombat.org/trader");
```

The following line of code gives an example of how a Name object can be used. The variable *element* is a `SOAPElement` object. This code creates a new `SOAPElement` object with the given name and adds it to *element*.

```
element.addChildElement(name);
```

The Name interface may be deprecated in a future release of SAAJ in favor of `javax.xml.namespace.QName`

See Also: [SOAPEnvelope.createName₈₇](#), [SOAPFactory.createName₉₈](#)

Member Summary	
Methods	
java.lang.String	getLocalName()₃₃ Gets the local name part of the XML name that this Name object represents.
java.lang.String	getPrefix()₃₃ Returns the prefix that was specified when this Name object was initialized.
java.lang.String	getQualifiedName()₃₃ Gets the namespace-qualified name of the XML name that this Name object represents.
java.lang.String	getURI()₃₄ Returns the URI of the namespace for the XML name that this Name object represents.

Methods

getLocalName()

```
public java.lang.String getLocalName()
```

Gets the local name part of the XML name that this Name object represents.

Returns: a string giving the local name

getPrefix()

```
public java.lang.String getPrefix()
```

Returns the prefix that was specified when this Name object was initialized. This prefix is associated with the namespace for the XML name that this Name object represents.

Returns: the prefix as a string

getQualifiedName()

```
public java.lang.String getQualifiedName()
```

Gets the namespace-qualified name of the XML name that this Name object represents.

Returns: the namespace-qualified name as a string

getURI()

```
public java.lang.String getURI()
```

Returns the URI of the namespace for the XML name that this Name object represents.

Returns: the URI as a string

2.9 Node

Declaration

```
public interface Node extends org.w3c.dom.Node
```

All Superinterfaces: `org.w3c.dom.Node`

All Known Subinterfaces: [Detail₁₇](#), [DetailEntry₂₀](#), [SOAPBody₄₆](#),
[SOAPBodyElement₅₄](#), [SOAPElement₆₇](#), [SOAPEnvelope₈₄](#), [SOAPFault₁₀₀](#),
[SOAPFaultElement₁₁₃](#), [SOAPHeader₁₁₅](#), [SOAPHeaderElement₁₂₃](#), [Text₁₄₆](#)

Description

A representation of a node (element) in an XML document. This interface extends the standard DOM Node interface with methods for getting and setting the value of a node, for getting and setting the parent of a node, and for removing a node.

Member Summary

Methods

void	detachNode()₃₆	Removes this Node object from the tree.
SOAPElement	getParentElement()₃₆	Returns the parent element of this Node object.
java.lang.String	getValue()₃₆	Returns the value of this node if this is a Text node or the value of the immediate child of this node otherwise.
void	recycleNode()₃₇	Notifies the implementation that this Node object is no longer being used by the application and that the implementation is free to reuse this object for nodes that may be created later.
void	setParentElement(SOAPElement parent)₃₇	Sets the parent of this Node object to the given SOAPElement object.
void	setValue(java.lang.String value)₃₇	If this is a Text node then this method will set its value, otherwise it sets the value of the immediate (Text) child of this node.

Inherited Member Summary

Fields inherited from interface Node

ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE

Methods inherited from interface Node

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getLocalName(), getNamespaceURI(), getNextSibling(), getNodeName(), getNodeType(), getNodeValue(), getOwnerDocument(), getParentNode(), getPrefix(), getPreviousSibling(), hasAttributes(), hasChildNodes(), insertBefore(Node, Node), isSupported(String, String), normalize(), removeChild(Node), replaceChild(Node, Node), setNodeValue(String), setPrefix(String)

Methods

detachNode()

```
public void detachNode()
```

Removes this Node object from the tree.

getParentElement()

```
public javax.xml.soap.SOAPElement67 getParentElement()
```

Returns the parent element of this Node object. This method can throw an `UnsupportedOperationException` if the tree is not kept in memory.

Returns: the `SOAPElement` object that is the parent of this Node object or `null` if this Node object is root

Throws:

`java.lang.UnsupportedOperationException` - if the whole tree is not kept in memory

See Also: [setParentElement\(SOAPElement\)](#)₃₇

getValue()

```
public java.lang.String getValue()
```

Returns the value of this node if this is a `Text` node or the value of the immediate child of this node otherwise. If there is an immediate child of this `Node` that it is a `Text` node then it's value will be returned. If there is more than one `Text` node then the value of the first `Text` Node will be returned. Otherwise `null` is returned.

Returns: a `String` with the text of this node if this is a `Text` node or the text contained by the first immediate child of this `Node` object that is a `Text` object if such a child exists; `null` otherwise.

recycleNode()

```
public void recycleNode()
```

Notifies the implementation that this `Node` object is no longer being used by the application and that the implementation is free to reuse this object for nodes that may be created later.

Calling the method `recycleNode` implies that the method `detachNode` has been called previously.

setParentElement(SOAPElement)

```
public void setParentElement(javax.xml.soap.SOAPElement67 parent)
    throws SOAPException
```

Sets the parent of this `Node` object to the given `SOAPElement` object.

Parameters:

`parent` - the `SOAPElement` object to be set as the parent of this `Node` object

Throws:

`SOAPException89` - if there is a problem in setting the parent to the given element

See Also: `getParentElement()36`

setValue(String)

```
public void setValue(java.lang.String value)
```

If this is a `Text` node then this method will set its value, otherwise it sets the value of the immediate (`Text`) child of this node. The value of the immediate child of this node can be set only if, there is one child node and that node is a `Text` node, or if there are no children in which case a child `Text` node will be created.

Parameters:

`value` - value to set on the `Text` node

Throws:

`java.lang.IllegalStateException` - if the node is not a `Text` node and either has more than one child node or has a child node that is not a `Text` node.

Since: SAAJ 1.2

2.10 SAAJMetaFactory

Declaration

```
public abstract class SAAJMetaFactory  
  
java.lang.Object  
|  
+-- javax.xml.soap.SAAJMetaFactory
```

Description

The access point for the implementation classes of the factories defined in the SAAJ API. The `newInstance` methods defined on factories `SOAPFactory` and `MessageFactory` in SAAJ 1.4 defer to instances of this class to do the actual object creation. The implementations of `newInstance()` methods (in `SOAPFactory` and `MessageFactory`) that existed in SAAJ 1.2 have been updated to also delegate to the `SAAJMetaFactory` when the SAAJ 1.2 defined lookup fails to locate the Factory implementation class name.

`SAAJMetaFactory` is a service provider interface and it uses similar lookup mechanism as other SAAJ factories to get actual instance:

- If a system property with name `javax.xml.soap.SAAJMetaFactory` exists then its value is assumed to be the fully qualified name of the implementation class. This phase of the look up enables per-JVM override of the SAAJ implementation.
- If a system property with name `javax.xml.soap.MetaFactory` exists then its value is assumed to be the fully qualified name of the implementation class. This property, defined by previous specifications (up to 1.3), is still supported, but it is strongly recommended to migrate to new property `javax.xml.soap.SAAJMetaFactory`.
- Use the configuration file "jaxm.properties". The file is in standard `Properties` format and typically located in the `conf` directory of the Java installation. It contains the fully qualified name of the implementation class with key `javax.xml.soap.SAAJMetaFactory`. If no such property is defined, again, property with key `javax.xml.soap.MetaFactory` is used. It is strongly recommended to migrate to new property `javax.xml.soap.SAAJMetaFactory`.
- Use the service-provider loading facilities, defined by the `ServiceLoader` class, to attempt to locate and load an implementation of the service using the default loading mechanism.
- Finally, if all the steps above fail, platform default implementation is used.

There are no public methods on this class.

Since: SAAJ 1.3

Member Summary	
Methods	
(package private) getInstance()	
static SAAJMetaFactory	Creates a new instance of a concrete SAAJMetaFactory object.
protected abstract newMessageFactory(java.lang.String protocol)	
MessageFactory	Creates a MessageFactory object for the given String protocol.
protected abstract newSOAPFactory(java.lang.String protocol)	
SOAPFactory	Creates a SOAPFactory object for the given String protocol.

Inherited Member Summary
Methods inherited from class Object
equals(Object), getClass(), hashCode(), notify(), notifyAll(), toString(), wait(long, int), wait(long, int), wait(long, int)

Methods

getInstance()

static SAAJMetaFactory getInstance() throws SOAPException

Creates a new instance of a concrete SAAJMetaFactory object. The SAAJMetaFactory is an SPI, it pulls the creation of the other factories together into a single place. Changing out the SAAJMetaFactory has the effect of changing out the entire SAAJ implementation. Service providers provide the name of their SAAJMetaFactory implementation.

This method uses the lookup procedure specified in javax.xml.soap to locate and load the SAAJMetaFactory class.

Returns: a concrete SAAJMetaFactory object

Throws: SOAPException_{g9} - if there is an error in creating the SAAJMetaFactory

newMessageFactory()

protected abstract MessageFactory newMessageFactory(java.lang.String protocol)

throws `SOAPException`

Creates a `MessageFactory` object for the given `String` protocol.

Parameters: `protocol` - a `String` indicating the protocol

Returns: a `MessageFactory`, not null

Throws: `SOAPException`₈₉ - if there is an error in creating the `MessageFactory`

See Also: `SOAP_1_1_PROTOCOL`₆₃, `SOAP_1_2_PROTOCOL`₆₃, `DYNAMIC_SOAP_PROTOCOL`₆₂

newSOAPFactory()

```
protected abstract SOAPFactory newSOAPFactory(java.lang.String protocol)
```

throws `SOAPException`

Creates a `SOAPFactory` object for the given `String` protocol.

Parameters: `protocol` - a `String` indicating the protocol

Returns: a `SOAPFactory`, not null

Throws: `SOAPException`₈₉ - if there is an error in creating the `SOAPFactory`

See Also: `SOAP_1_1_PROTOCOL`₆₃, `SOAP_1_2_PROTOCOL`₆₃,
`DYNAMIC_SOAP_PROTOCOL`₆₂

2.11 SAAJResult

Declaration

```
public class SAAJResult extends javax.xml.transform.dom.DOMResult
```

```
java.lang.Object
|
+--javax.xml.transform.dom.DOMResult
|
+--javax.xml.soap.SAAJResult
```

All Implemented Interfaces:
javax.xml.transform.Result

Description

Acts as a holder for the results of a JAXP transformation or a JAXB marshalling, in the form of a SAAJ tree. These results should be accessed by using the [getResult\(\)](#)⁴⁵ method. The `javax.xml.transform.dom.DOMResult.getNode()` method should be avoided in almost all cases.

Since: SAAJ 1.3

Member Summary

Constructors

- [SAAJResult\(\)](#)⁴³
Creates a SAAJResult that will present results in the form of a SAAJ tree that supports the default (SOAP 1.1) protocol.
- [SAAJResult\(SOAPElement rootNode\)](#)⁴⁴
Creates a SAAJResult that will write the results as a child node of the SOAPElement specified.
- [SAAJResult\(SOAPMessage message\)](#)⁴⁴
Creates a SAAJResult that will write the results into the SOAPPart of the supplied SOAPMessage.
- [SAAJResult\(java.lang.String protocol\)](#)⁴⁴
Creates a SAAJResult that will present results in the form of a SAAJ tree that supports the specified protocol.

Member Summary
Methods <div>Node <code>getResult()</code>₄₅</div>

Inherited Member Summary
Fields inherited from class DOMResult FEATURE
Fields inherited from interface Result PI_DISABLE_OUTPUT_ESCAPING, PI_ENABLE_OUTPUT_ESCAPING
Methods inherited from class DOMResult getNode(), getSystemId(), setNode(Node), setSystemId(String)
Methods inherited from class Object equals(Object), getClass(), hashCode(), notify(), notifyAll(), toString(), wait(long, int), wait(long, int), wait(long, int)

Constructors

SAAJResult()

```
public SAAJResult()
    throws SOAPException
```

Creates a SAAJResult that will present results in the form of a SAAJ tree that supports the default (SOAP 1.1) protocol.

This kind of SAAJResult is meant for use in situations where the results will be used as a parameter to a method that takes a parameter whose type, such as SOAPElement, is drawn from the SAAJ API. When used in a transformation, the results are populated into the SOAPPart of a SOAPMessage that is created internally. The SOAPPart returned by `javax.xml.transform.dom.DOMResult.getNode()` is not guaranteed to be well-formed.

Throws:

`SOAPException`₈₉ - if there is a problem creating a SOAPMessage

Since: SAAJ 1.3

SAAJResult(SOAPElement)

```
public SAAJResult(javax.xml.soap.SOAPElement67 rootNode)
```

Creates a SAAJResult that will write the results as a child node of the SOAPElement specified. In the normal case these results will be written using DOM APIs and as a result may invalidate the structure of the SAAJ tree. This kind of SAAJResult should only be used when the validity of the incoming data can be guaranteed by means outside of the SAAJ specification.

Parameters:

rootNode - the root to which the results will be appended

Since: SAAJ 1.3

SAAJResult(SOAPMessage)

```
public SAAJResult(javax.xml.soap.SOAPMessage128 message)
```

Creates a SAAJResult that will write the results into the SOAPPart of the supplied SOAPMessage. In the normal case these results will be written using DOM APIs and, as a result, the finished SOAPPart will not be guaranteed to be well-formed unless the data used to create it is also well formed. When used in a transformation the validity of the SOAPMessage after the transformation can be guaranteed only by means outside SAAJ specification.

Parameters:

message - the message whose SOAPPart will be populated as a result of some transformation or marshalling operation

Since: SAAJ 1.3

SAAJResult(String)

```
public SAAJResult(java.lang.String protocol)
    throws SOAPException
```

Creates a SAAJResult that will present results in the form of a SAAJ tree that supports the specified protocol. The DYNAMIC_SOAP_PROTOCOL is ambiguous in this context and will cause this constructor to throw a UnsupportedOperationException.

This kind of SAAJResult is meant for use in situations where the results will be used as a parameter to a method that takes a parameter whose type, such as SOAPElement, is drawn from the SAAJ API. When used in a transformation the results are populated into the SOAPPart of a SOAPMessage that is created internally. The SOAPPart returned by `javax.xml.transform.dom.DOMResult.getNode()` is not guaranteed to be well-formed.

Parameters:

`protocol` -the name of the SOAP protocol that the resulting SAAJ tree should support

Throws:

`SOAPException89` - if a SOAPMessage supporting the specified protocol cannot be created

Since: SAAJ 1.3

Methods

getResult()

```
public javax.xml.soap.Node35 getResult()
```

Returns: the resulting Tree that was created under the specified root Node.

Since: SAAJ 1.3

2.12 SOAPBody

Declaration

```
public interface SOAPBody extends SOAPElement67
```

All Superinterfaces: `org.w3c.dom.Element`, `org.w3c.dom.Node`, `Node35`, `SOAPElement67`

Description

An object that represents the contents of the SOAP body element in a SOAP message. A SOAP body element consists of XML data that affects the way the application-specific content is processed.

A SOAPBody object contains SOAPBodyElement objects, which have the content for the SOAP body. A SOAPFault object, which carries status and/or error information, is an example of a SOAPBodyElement object.

See Also: `SOAPFault100`

Member Summary	
Methods	
SOAPBodyElement	<code>addBodyElement(Name name)₄₈</code> Creates a new SOAPBodyElement object with the specified name and adds it to this SOAPBody object.
SOAPBodyElement	<code>addBodyElement(javax.xml.namespace.QName qname)₄₈</code> Creates a new SOAPBodyElement object with the specified QName and adds it to this SOAPBody object.
SOAPBodyElement	<code>addDocument(org.w3c.dom.Document document)₄₉</code> Adds the root node of the DOM <code>org.w3c.dom.Document</code> to this SOAPBody object.
SOAPFault	<code>addFault()₄₉</code> Creates a new SOAPFault object and adds it to this SOAPBody object.
SOAPFault	<code>addFault(Name faultCode, java.lang.String faultString)₅₀</code> Creates a new SOAPFault object and adds it to this SOAPBody object.

Member Summary	
SOAPFault	addFault (Name faultCode, java.lang.String faultString, java.util.Locale locale) ⁵⁰ Creates a new SOAPFault object and adds it to this SOAPBody object.
SOAPFault	addFault (javax.xml.namespace.QName faultCode, java.lang.String faultString) ⁵¹ Creates a new SOAPFault object and adds it to this SOAPBody object.
SOAPFault	addFault (javax.xml.namespace.QName faultCode, java.lang.String faultString, java.util.Locale locale) ⁵² Creates a new SOAPFault object and adds it to this SOAPBody object.
org.w3c.dom.Document	extractContentAsDocument () ⁵² Creates a new DOM org.w3c.dom.Document and sets the first child of this SOAPBody as it's document element.
SOAPFault	getFault () ⁵³ Returns the SOAPFault object in this SOAPBody object.
boolean	hasFault () ⁵³ Indicates whether a SOAPFault object exists in this SOAPBody object.

Inherited Member Summary
Fields inherited from interface Node ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE
Methods inherited from interface Element getAttribute(String), getAttributeNS(String, String), getAttributeNode(String), getAttributeNodeNS(String, String), getElementsByTagName(String), getElementsByTagNameNS(String, String), getTagName(), hasAttribute(String), hasAttributeNS(String, String), removeAttribute(String), removeAttributeNS(String, String), removeAttributeNode(Attr), setAttribute(String, String), setAttributeNS(String, String, String), setAttributeNode(Attr), setAttributeNodeNS(Attr)
Methods inherited from interface Node₃₅ detachNode() ₃₆ , getParentElement() ₃₆ , getValue() ₃₆ , recycleNode() ₃₇ , setParentElement(SOAPElement) ₃₇ , setValue(String) ₃₇
Methods inherited from interface Node

Inherited Member Summary

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(),
getFirstChild(), getLastChild(), getLocalName(), getNamespaceURI(), getNextSibling(),
getNodeName(), getNodeType(), getNodeValue(), getOwnerDocument(), getParentNode(),
getPrefix(), getPreviousSibling(), hasAttributes(), hasChildNodes(),
insertBefore(Node, Node), isSupported(String, String), normalize(),
removeChild(Node), replaceChild(Node, Node), setNodeValue(String), setPrefix(String)

Methods inherited from interface [SOAPElement](#)₆₇

addAttribute(Name, String)₇₀, addChildElement(SOAPElement)₇₂,
addChildElement(SOAPElement)₇₂, addChildElement(SOAPElement)₇₂,
addChildElement(SOAPElement)₇₂, addChildElement(SOAPElement)₇₂,
addNamespaceDeclaration(String, String)₇₃, addTextNode(String)₇₄,
getAllAttributes()₇₄, getAttributeValue(Name)₇₅, getChildElements(Name)₇₆,
getChildElements(Name)₇₆, getElementName()₇₇, getEncodingStyle()₇₇,
getNamespacePrefixes()₇₇, getNamespaceURI(String)₇₈, getVisibleNamespacePrefixes()₇₈,
removeAttribute(Name)₇₈, removeContents()₇₉, removeNamespaceDeclaration(String)₇₉,
setEncodingStyle(String)₇₉

Methods

addBodyElement(Name)

```
public javax.xml.soap.SOAPBodyElement54 addBodyElement(javax.xml.soap.Name32 name)
    throws SOAPException
```

Creates a new SOAPBodyElement object with the specified name and adds it to this SOAPBody object.

Parameters:

name - a Name object with the name for the new SOAPBodyElement object

Returns: the new SOAPBodyElement object

Throws:

[SOAPException](#)₈₉ - if a SOAP error occurs

addBodyElement(QName)

```
public javax.xml.soap.SOAPBodyElement54 addBodyElement(javax.xml.namespace.QName
    qname)
    throws SOAPException
```

Creates a new SOAPBodyElement object with the specified QName and adds it to this SOAPBody object.

Parameters:

qname - a QName object with the qname for the new SOAPBodyElement object

Returns: the new SOAPBodyElement object

Throws:

SOAPException₈₉ - if a SOAP error occurs

Since: SAAJ 1.3

See Also: addBodyElement (Name)₄₈

addDocument(Document)

```
public javax.xml.soap.SOAPBodyElement54 addDocument(org.w3c.dom.Document document)
    throws SOAPException
```

Adds the root node of the DOM org.w3c.dom.Document to this SOAPBody object.

Calling this method invalidates the document parameter. The client application should discard all references to this Document and its contents upon calling addDocument. The behavior of an application that continues to use such references is undefined.

Parameters:

document - the Document object whose root node will be added to this SOAPBody.

Returns: the SOAPBodyElement that represents the root node that was added.

Throws:

SOAPException₈₉ - if the Document cannot be added

Since: SAAJ 1.2

addFault()

```
public javax.xml.soap.SOAPFault100 addFault()
    throws SOAPException
```

Creates a new SOAPFault object and adds it to this SOAPBody object. The new SOAPFault will have default values set for the mandatory child elements. The type of the SOAPFault will be a SOAP 1.1 or a SOAP 1.2 SOAPFault depending on the protocol specified while creating the MessageFactory instance

A SOAPBody may contain at most one SOAPFault child element

Returns: the new SOAPFault object

Throws:

SOAPException₈₉ - if there is a SOAP error

addFault(Name, String)

```
public javax.xml.soap.SOAPFault100 addFault(javax.xml.soap.Name32 faultCode,  
    java.lang.String faultString)  
    throws SOAPException
```

Creates a new SOAPFault object and adds it to this SOAPBody object. The type of the SOAPFault will be a SOAP 1.1 or a SOAP 1.2 SOAPFault depending on the protocol specified while creating the MessageFactory instance.

For SOAP 1.2 the faultCode parameter is the value of the *Fault/Code/Value* element and the faultString parameter is the value of the *Fault/Reason/Text* element. For SOAP 1.1 the faultCode parameter is the value of the *faultcode* element and the faultString parameter is the value of the *faultstring* element.

In case of a SOAP 1.2 fault, the default value for the mandatory `xml:lang` attribute on the *Fault/Reason/Text* element will be set to `java.util.Locale.getDefault()`

A SOAPBody may contain at most one SOAPFault child element

Parameters:

faultCode - a Name object giving the fault code to be set; must be one of the fault codes defined in the SOAP 1.1 specification and of type QName

faultString - a String giving an explanation of the fault

Returns: the new SOAPFault object

Throws:

SOAPException₈₉ - if there is a SOAP error

Since: SAAJ 1.2

See Also: SOAPFault.setFaultCode(Name)₁₀₉,
SOAPFault.setFaultString(String)₁₁₁

addFault(Name, String, Locale)

```
public javax.xml.soap.SOAPFault100 addFault(javax.xml.soap.Name32 faultCode,  
    java.lang.String faultString, java.util.Locale locale)  
    throws SOAPException
```

Creates a new SOAPFault object and adds it to this SOAPBody object. The type of the SOAPFault will be a SOAP 1.1 or a SOAP 1.2 SOAPFault depending on the protocol specified while creating the MessageFactory instance.

For SOAP 1.2 the faultCode parameter is the value of the *Fault/Code/Value* element and the faultString parameter is the value of the *Fault/Reason/Text* element. For SOAP 1.1 the faultCode parameter is the value of the *faultcode* element and the faultString parameter is the value of the *faultstring* element.

A `SOAPBody` may contain at most one `SOAPFault` child element

Parameters:

`faultCode` - a `Name` object giving the fault code to be set; must be one of the fault codes defined in the version of SOAP specification in use.

`faultString` - a `String` giving an explanation of the fault

`locale` - a `Locale` object indicating the native language of the `faultString`

Returns: the new `SOAPFault` object

Throws:

`SOAPException`₈₉ - if there is a SOAP error

Since: SAAJ 1.2

See Also: `SOAPFault.setFaultCode(Name)`₁₀₉,
`SOAPFault.setFaultString(String)`₁₁₁

addFault(QName, String)

```
public javax.xml.soap.SOAPFault100 addFault(javax.xml.namespace.QName faultCode,  
      java.lang.String faultString)  
      throws SOAPException
```

Creates a new `SOAPFault` object and adds it to this `SOAPBody` object. The type of the `SOAPFault` will be a SOAP 1.1 or a SOAP 1.2 `SOAPFault` depending on the protocol specified while creating the `MessageFactory` instance.

For SOAP 1.2 the `faultCode` parameter is the value of the `Fault/Code/Value` element and the `faultString` parameter is the value of the `Fault/Reason/Text` element. For SOAP 1.1 the `faultCode` parameter is the value of the `faultcode` element and the `faultString` parameter is the value of the `faultstring` element.

In case of a SOAP 1.2 fault, the default value for the mandatory `xml:lang` attribute on the `Fault/Reason/Text` element will be set to `java.util.Locale.getDefault()`

A `SOAPBody` may contain at most one `SOAPFault` child element

Parameters:

`faultCode` - a `QName` object giving the fault code to be set; must be one of the fault codes defined in the version of SOAP specification in use

`faultString` - a `String` giving an explanation of the fault

Returns: the new `SOAPFault` object

Throws:

`SOAPException`₈₉ - if there is a SOAP error

Since: SAAJ 1.3

See Also: `SOAPFault.setFaultCode(Name)`₁₀₉,
`SOAPFault.setFaultString(String)`₁₁₁, `addFault(Name, String)`₅₀

addFault(QName, String, Locale)

```
public javax.xml.soap.SOAPFault100 addFault(javax.xml.namespace.QName faultCode,  
    java.lang.String faultString, java.util.Locale locale)  
    throws SOAPException
```

Creates a new SOAPFault object and adds it to this SOAPBody object. The type of the SOAPFault will be a SOAP 1.1 or a SOAP 1.2 SOAPFault depending on the protocol specified while creating the MessageFactory instance.

For SOAP 1.2 the `faultCode` parameter is the value of the *Fault/Code/Value* element and the `faultString` parameter is the value of the *Fault/Reason/Text* element. For SOAP 1.1 the `faultCode` parameter is the value of the `faultcode` element and the `faultString` parameter is the value of the `faultstring` element.

A SOAPBody may contain at most one SOAPFault child element.

Parameters:

`faultCode` - a QName object giving the fault code to be set; must be one of the fault codes defined in the version of SOAP specification in use.

`faultString` - a String giving an explanation of the fault

`locale` - a Locale object indicating the native language of the `faultString`

Returns: the new SOAPFault object

Throws:

`SOAPException`₈₉ - if there is a SOAP error

Since: SAAJ 1.3

See Also: `SOAPFault.setFaultCode(Name)`₁₀₉,
`SOAPFault.setFaultString(String)`₁₁₁, `addFault(Name, String, Locale)`₅₀

extractContentAsDocument()

```
public org.w3c.dom.Document extractContentAsDocument()  
    throws SOAPException
```

Creates a new DOM `org.w3c.dom.Document` and sets the first child of this SOAPBody as it's document element. The child `SOAPElement` is removed as part of the process.

Returns: the `org.w3c.dom.Document` representation of the SOAPBody content.

Throws:

`SOAPException`₈₉ - if there is not exactly one child `SOAPElement` of the SOAPBody.

getFault()

```
public javax.xml.soap.SOAPFault100 getFault()
```

Returns the SOAPFault object in this SOAPBody object.

Returns: the SOAPFault object in this SOAPBody object if present, null otherwise

hasFault()

```
public boolean hasFault()
```

Indicates whether a SOAPFault object exists in this SOAPBody object.

Returns: true if a SOAPFault object exists in this SOAPBody object; false otherwise

2.13 SOAPBodyElement

Declaration

```
public interface SOAPBodyElement extends SOAPElement67
```

All Superinterfaces: org.w3c.dom.Element, org.w3c.dom.Node, Node₃₅, SOAPElement₆₇

All Known Subinterfaces: SOAPFault₁₀₀

Description

A SOAPBodyElement object represents the contents in a SOAPBody object. The SOAPFault interface is a SOAPBodyElement object that has been defined.

A new SOAPBodyElement object can be created and added to a SOAPBody object with the SOAPBody method addBodyElement. In the following line of code, sb is a SOAPBody object, and myName is a Name object.

```
SOAPBodyElement sbe = sb.addBodyElement(myName);
```

Inherited Member Summary
<p>Fields inherited from interface Node</p> <p>ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE</p>
<p>Methods inherited from interface Element</p> <p>getAttribute(String), getAttributeNS(String, String), getAttributeNode(String), getAttributeNodeNS(String, String), getElementsByTagName(String), getElementsByTagNameNS(String, String), getTagName(), hasAttribute(String), hasAttributeNS(String, String), removeAttribute(String), removeAttributeNS(String, String), removeAttributeNode(Attr), setAttribute(String, String), setAttributeNS(String, String, String), setAttributeNode(Attr), setAttributeNodeNS(Attr)</p>

Inherited Member Summary

Methods inherited from interface [Node](#)₃₅

[detachNode\(\)](#)₃₆, [getParentElement\(\)](#)₃₆, [getValue\(\)](#)₃₆, [recycleNode\(\)](#)₃₇,
[setParentElement\(SOAPElement\)](#)₃₇, [setValue\(String\)](#)₃₇

Methods inherited from interface [Node](#)

[appendChild\(Node\)](#), [cloneNode\(boolean\)](#), [getAttributes\(\)](#), [getChildNodes\(\)](#),
[getFirstChild\(\)](#), [getLastChild\(\)](#), [getLocalName\(\)](#), [getNamespaceURI\(\)](#), [getNextSibling\(\)](#),
[getNodeName\(\)](#), [getNodeType\(\)](#), [getNodeValue\(\)](#), [getOwnerDocument\(\)](#), [getParentNode\(\)](#),
[getPrefix\(\)](#), [getPreviousSibling\(\)](#), [hasAttributes\(\)](#), [hasChildNodes\(\)](#),
[insertBefore\(Node, Node\)](#), [isSupported\(String, String\)](#), [normalize\(\)](#),
[removeChild\(Node\)](#), [replaceChild\(Node, Node\)](#), [setNodeValue\(String\)](#), [setPrefix\(String\)](#)

Methods inherited from interface [SOAPElement](#)₆₇

[addAttribute\(Name, String\)](#)₇₀, [addChildElement\(SOAPElement\)](#)₇₂,
[addChildElement\(SOAPElement\)](#)₇₂, [addChildElement\(SOAPElement\)](#)₇₂,
[addChildElement\(SOAPElement\)](#)₇₂, [addChildElement\(SOAPElement\)](#)₇₂,
[addNamespaceDeclaration\(String, String\)](#)₇₃, [addTextNode\(String\)](#)₇₄,
[getAllAttributes\(\)](#)₇₄, [getAttributeValue\(Name\)](#)₇₅, [getChildElements\(Name\)](#)₇₆,
[getChildElements\(Name\)](#)₇₆, [getElementName\(\)](#)₇₇, [getEncodingStyle\(\)](#)₇₇,
[getNamespacePrefixes\(\)](#)₇₇, [getNamespaceURI\(String\)](#)₇₈, [getVisibleNamespacePrefixes\(\)](#)₇₈,
[removeAttribute\(Name\)](#)₇₈, [removeContents\(\)](#)₇₉, [removeNamespaceDeclaration\(String\)](#)₇₉,
[setEncodingStyle\(String\)](#)₇₉

2.14 SOAPConnection

Declaration

```
public abstract class SOAPConnection
```

```
java.lang.Object
|
+-- javax.xml.soap.SOAPConnection
```

Description

A point-to-point connection that a client can use for sending messages directly to a remote party (represented by a URL, for instance).

The SOAPConnection class is optional. Some implementations may not implement this interface in which case the call to SOAPConnectionFactory.newInstance() (see below) will throw an UnsupportedOperationException.

A client can obtain a SOAPConnection object using a SOAPConnectionFactory₅₉ object as in the following example:

```
SOAPConnectionFactory factory = SOAPConnectionFactory.newInstance();
SOAPConnection con = factory.createConnection();
```

A SOAPConnection object can be used to send messages directly to a URL following the request/response paradigm. That is, messages are sent using the method call, which sends the message and then waits until it gets a reply.

Member Summary	
Constructors	
	<code>SOAPConnection()</code> ₅₇
Methods	
<code>abstract SOAPMessage</code>	<code>call(SOAPMessage request, java.lang.Object to)</code> ₅₇ Sends the given message to the specified endpoint and blocks until it has returned the response.
<code>abstract void</code>	<code>close()</code> ₅₇ Closes this SOAPConnection object.

Member Summary

SOAPMessage [get\(java.lang.Object to\)](#)⁵⁸
Gets a message from a specific endpoint and blocks until it receives,

Inherited Member Summary

Methods inherited from class Object

clone(), equals(Object), finalize(), getClass(), hashCode(), notify(), notifyAll(), toString(), wait(long, int), wait(long, int), wait(long, int)

Constructors

SOAPConnection()

```
public SOAPConnection()
```

Methods

call(SOAPMessage, Object)

```
public abstract javax.xml.soap.SOAPMessage128 call(javax.xml.soap.SOAPMessage128  
    request, java.lang.Object to)  
    throws SOAPException
```

Sends the given message to the specified endpoint and blocks until it has returned the response.

Parameters:

request - the SOAPMessage object to be sent

to - an Object that identifies where the message should be sent. It is required to support Objects of type java.lang.String, java.net.URL, and when JAXM is present javax.xml.messaging.URLEndpoint

Returns: the SOAPMessage object that is the response to the message that was sent

Throws:

[SOAPException](#)⁸⁹ - if there is a SOAP error

close()

```
public abstract void close()  
    throws SOAPException
```


Closes this `SOAPConnection` object.

Throws:

`SOAPException89` - if there is a SOAP error

get(Object)

```
public javax.xml.soap.SOAPMessage128 get(java.lang.Object to)
    throws SOAPException
```

Gets a message from a specific endpoint and blocks until it receives,

Parameters:

`to` - an `Object` that identifies where the request should be sent. Objects of type `java.lang.String` and `java.net.URL` must be supported.

Returns: the `SOAPMessage` object that is the response to the get message request

Throws:

`SOAPException89` - if there is a SOAP error

Since: SAAJ 1.3

2.15 SOAPConnectionFactory

Declaration

```
public abstract class SOAPConnectionFactory

java.lang.Object
|
+--javax.xml.soap.SOAPConnectionFactory
```

Description

A factory for creating SOAPConnection objects. Implementation of this class is optional. If SOAPConnectionFactory.newInstance() throws an UnsupportedOperationException then the implementation does not support the SAAJ communication infrastructure. Otherwise SOAPConnection₅₆ objects can be created by calling createConnection() on the newly created SOAPConnectionFactory object.

Member Summary

Constructors

SOAPConnectionFactory() ₆₀

Methods

abstract	createConnection() ₆₀	
SOAPConnection		Create a new SOAPConnection.
static	newInstance() ₆₀	
SOAPConnectionFactory		Creates an instance of the default SOAPConnectionFactory object.

Inherited Member Summary

Methods inherited from class Object

clone(), equals(Object), finalize(), getClass(), hashCode(), notify(), notifyAll(), toString(), wait(long, int), wait(long, int), wait(long, int)

Constructors

SOAPConnectionFactory()

```
public SOAPConnectionFactory()
```

Methods

createConnection()

```
public abstract javax.xml.soap.SOAPConnection56 createConnection()  
    throws SOAPException
```

Create a new SOAPConnection.

Returns: the new SOAPConnection object.

Throws:

[SOAPException₈₉](#) - if there was an exception creating the SOAPConnection object.

newInstance()

```
public static javax.xml.soap.SOAPConnectionFactory59 newInstance()  
    throws SOAPException, UnsupportedOperationException
```

Creates an instance of the default SOAPConnectionFactory object. This method uses the lookup procedure specified in `javax.xml.soap` to locate and load the SOAPConnectionFactory class.

Returns: a new instance of a default SOAPConnectionFactory object

Throws:

[SOAPException₈₉](#) - if there was an error creating the SOAPConnectionFactory
`java.lang.UnsupportedOperationException` - if newInstance is not supported.

2.16 SOAPConstants

Declaration

```
public interface SOAPConstants
```

Description

The definition of constants pertaining to SOAP protocol.

Member Summary

Fields

static	DEFAULT_SOAP_PROTOCOL ₆₂	
java.lang.String		The default protocol: SOAP 1.1 for backwards compatibility.
static	DYNAMIC_SOAP_PROTOCOL ₆₂	
java.lang.String		Used to create MessageFactory instances that create SOAPMessages whose concrete type is based on the Content-Type MIME header passed to the createMessage method.
static	SOAP_1_1_CONTENT_TYPE ₆₃	
java.lang.String		The media type of the Content-Type MIME header in SOAP 1.1.
static	SOAP_1_1_PROTOCOL ₆₃	
java.lang.String		Used to create MessageFactory instances that create SOAPMessages whose behavior supports the SOAP 1.1 specification.
static	SOAP_1_2_CONTENT_TYPE ₆₃	
java.lang.String		The media type of the Content-Type MIME header in SOAP 1.2.
static	SOAP_1_2_PROTOCOL ₆₃	
java.lang.String		Used to create MessageFactory instances that create SOAPMessages whose behavior supports the SOAP 1.2 specification
static	SOAP_DATAENCODINGUNKNOWN_FAULT ₆₃	
javax.xml.namespace.QName		SOAP 1.2 DataEncodingUnknown Fault
static	SOAP_ENV_PREFIX ₆₃	
java.lang.String		The default namespace prefix for http://www.w3.org/2003/05/soap-envelope
static	SOAP_MUSTUNDERSTAND_FAULT ₆₄	
javax.xml.namespace.QName		SOAP 1.2 MustUnderstand Fault
static	SOAP_RECEIVER_FAULT ₆₄	
javax.xml.namespace.QName		SOAP 1.2 Receiver Fault

Member Summary

<code>javax.xml.namespace.QName</code>	<code>static SOAP_SENDER_FAULT⁶⁴</code>	SOAP 1.2 Sender Fault
<code>javax.xml.namespace.QName</code>	<code>static SOAP_VERSIONMISMATCH_FAULT⁶⁴</code>	SOAP 1.2 VersionMismatch Fault
<code>java.lang.String</code>	<code>static URI_NS_SOAP_1_1_ENVELOPE⁶⁴</code>	The namespace identifier for the SOAP 1.1 envelope.
<code>java.lang.String</code>	<code>static URI_NS_SOAP_1_2_ENCODING⁶⁴</code>	The namespace identifier for the SOAP 1.2 encoding.
<code>java.lang.String</code>	<code>static URI_NS_SOAP_1_2_ENVELOPE⁶⁵</code>	The namespace identifier for the SOAP 1.2 envelope.
<code>java.lang.String</code>	<code>static URI_NS_SOAP_ENCODING⁶⁵</code>	The namespace identifier for the SOAP 1.1 encoding.
<code>java.lang.String</code>	<code>static URI_NS_SOAP_ENVELOPE⁶⁵</code>	The namespace identifier for the SOAP 1.1 envelope.
<code>java.lang.String</code>	<code>static URI_SOAP_1_2_ROLE_NEXT⁶⁵</code>	The URI identifying the next application processing a SOAP request as the intended role for a SOAP 1.2 header entry (see section 2.2 of part 1 of the SOAP 1.2 specification).
<code>java.lang.String</code>	<code>static URI_SOAP_1_2_ROLE_NONE⁶⁵</code>	The URI specifying the role None in SOAP 1.2.
<code>java.lang.String</code>	<code>static URI_SOAP_1_2_ROLE_ULTIMATE_RECEIVER⁶⁵</code>	The URI identifying the ultimate receiver of the SOAP 1.2 message.
<code>java.lang.String</code>	<code>static URI_SOAP_ACTOR_NEXT⁶⁶</code>	The URI identifying the next application processing a SOAP request as the intended actor for a SOAP 1.1 header entry (see section 4.2.2 of the SOAP 1.1 specification).

Fields

DEFAULT_SOAP_PROTOCOL

```
public static final java.lang.String DEFAULT_SOAP_PROTOCOL
```

The default protocol: SOAP 1.1 for backwards compatibility.

Since: SAAJ 1.3

DYNAMIC_SOAP_PROTOCOL

```
public static final java.lang.String DYNAMIC_SOAP_PROTOCOL
```

Used to create `MessageFactory` instances that create `SOAPMessages` whose concrete type is based on the Content-Type MIME header passed to the `createMessage` method. If no Content-Type header is passed then the `createMessage` may throw an

`IllegalArgumentException` or, in the case of the no argument version of `createMessage`, an `UnsupportedOperationException`.

Since: SAAJ 1.3

SOAP_1_1_CONTENT_TYPE

```
public static final java.lang.String SOAP_1_1_CONTENT_TYPE
```

The media type of the Content-Type MIME header in SOAP 1.1.

Since: SAAJ 1.3

SOAP_1_1_PROTOCOL

```
public static final java.lang.String SOAP_1_1_PROTOCOL
```

Used to create `MessageFactory` instances that create `SOAPMessages` whose behavior supports the SOAP 1.1 specification.

Since: SAAJ 1.3

SOAP_1_2_CONTENT_TYPE

```
public static final java.lang.String SOAP_1_2_CONTENT_TYPE
```

The media type of the Content-Type MIME header in SOAP 1.2.

Since: SAAJ 1.3

SOAP_1_2_PROTOCOL

```
public static final java.lang.String SOAP_1_2_PROTOCOL
```

Used to create `MessageFactory` instances that create `SOAPMessages` whose behavior supports the SOAP 1.2 specification

Since: SAAJ 1.3

SOAP_DATAENCODINGUNKNOWN_FAULT

```
public static final javax.xml.namespace.QName SOAP_DATAENCODINGUNKNOWN_FAULT
```

SOAP 1.2 `DataEncodingUnknown` Fault

Since: SAAJ 1.3

SOAP_ENV_PREFIX

```
public static final java.lang.String SOAP_ENV_PREFIX
```

The default namespace prefix for <http://www.w3.org/2003/05/soap-envelope>

Since: SAAJ 1.3

SOAP_MUSTUNDERSTAND_FAULT

```
public static final javax.xml.namespace.QName SOAP_MUSTUNDERSTAND_FAULT
```

SOAP 1.2 MustUnderstand Fault

Since: SAAJ 1.3

SOAP_RECEIVER_FAULT

```
public static final javax.xml.namespace.QName SOAP_RECEIVER_FAULT
```

SOAP 1.2 Receiver Fault

Since: SAAJ 1.3

SOAP_SENDER_FAULT

```
public static final javax.xml.namespace.QName SOAP_SENDER_FAULT
```

SOAP 1.2 Sender Fault

Since: SAAJ 1.3

SOAP_VERSIONMISMATCH_FAULT

```
public static final javax.xml.namespace.QName SOAP_VERSIONMISMATCH_FAULT
```

SOAP 1.2 VersionMismatch Fault

Since: SAAJ 1.3

URI_NS_SOAP_1_1_ENVELOPE

```
public static final java.lang.String URI_NS_SOAP_1_1_ENVELOPE
```

The namespace identifier for the SOAP 1.1 envelope.

Since: SAAJ 1.3

URI_NS_SOAP_1_2_ENCODING

```
public static final java.lang.String URI_NS_SOAP_1_2_ENCODING
```

The namespace identifier for the SOAP 1.2 encoding.

Since: SAAJ 1.3

URI_NS_SOAP_1_2_ENVELOPE

```
public static final java.lang.String URI_NS_SOAP_1_2_ENVELOPE
```

The namespace identifier for the SOAP 1.2 envelope.

Since: SAAJ 1.3

URI_NS_SOAP_ENCODING

```
public static final java.lang.String URI_NS_SOAP_ENCODING
```

The namespace identifier for the SOAP 1.1 encoding. An attribute named `encodingStyle` in the `URI_NS_SOAP_ENVELOPE` namespace and set to the value `URI_NS_SOAP_ENCODING` can be added to an element to indicate that it is encoded using the rules in section 5 of the SOAP 1.1 specification.

URI_NS_SOAP_ENVELOPE

```
public static final java.lang.String URI_NS_SOAP_ENVELOPE
```

The namespace identifier for the SOAP 1.1 envelope. All SOAP elements in this namespace are defined by the SOAP 1.1 specification.

URI_SOAP_1_2_ROLE_NEXT

```
public static final java.lang.String URI_SOAP_1_2_ROLE_NEXT
```

The URI identifying the next application processing a SOAP request as the intended role for a SOAP 1.2 header entry (see section 2.2 of part 1 of the SOAP 1.2 specification).

Since: SAAJ 1.3

URI_SOAP_1_2_ROLE_NONE

```
public static final java.lang.String URI_SOAP_1_2_ROLE_NONE
```

The URI specifying the role None in SOAP 1.2.

Since: SAAJ 1.3

URI_SOAP_1_2_ROLE_ULTIMATE_RECEIVER

```
public static final java.lang.String URI_SOAP_1_2_ROLE_ULTIMATE_RECEIVER
```

The URI identifying the ultimate receiver of the SOAP 1.2 message.

Since: SAAJ 1.3

URI_SOAP_ACTOR_NEXT

```
public static final java.lang.String URI_SOAP_ACTOR_NEXT
```

The URI identifying the next application processing a SOAP request as the intended actor for a SOAP 1.1 header entry (see section 4.2.2 of the SOAP 1.1 specification).

This value can be passed to

```
SOAPHeader.examineMustUnderstandHeaderElements(String)121,  
SOAPHeader.examineHeaderElements(String)120 and  
SOAPHeader.extractHeaderElements(String)121
```

2.17 SOAPElement

Declaration

```
public interface SOAPElement extends Node35, org.w3c.dom.Element
```

All Superinterfaces: org.w3c.dom.Element, org.w3c.dom.Node, Node₃₅

All Known Subinterfaces: Detail₁₇, DetailEntry₂₀, SOAPBody₄₆, SOAPBodyElement₅₄, SOAPEnvelope₈₄, SOAPFault₁₀₀, SOAPFaultElement₁₁₃, SOAPHeader₁₁₅, SOAPHeaderElement₁₂₃

Description

An object representing an element of a SOAP message that is allowed but not specifically prescribed by a SOAP specification. This interface serves as the base interface for those objects that are specifically prescribed by a SOAP specification.

Methods in this interface that are required to return SAAJ specific objects may “silently” replace nodes in the tree as required to successfully return objects of the correct type. See [getChildElements\(\)](#)₇₅ and [javax.xml.soap](#) (package-summary.html#package_description) for details.

Member Summary	
Methods	
SOAPElement	addAttribute(Name name, java.lang.String value) ₇₀ Adds an attribute with the specified name and value to this SOAPElement object.
SOAPElement	addAttribute(javax.xml.namespace.QName qname, java.lang.String value) ₇₀ Adds an attribute with the specified name and value to this SOAPElement object.
SOAPElement	addChildElement(Name name) ₇₁ Creates a new SOAPElement object initialized with the given Name object and adds the new element to this SOAPElement object.
SOAPElement	addChildElement(javax.xml.namespace.QName qname) ₇₁ Creates a new SOAPElement object initialized with the given QName object and adds the new element to this SOAPElement object.

Member Summary

SOAPElement	addChildElement(SOAPElement element) ⁷²	Add a SOAPElement as a child of this SOAPElement instance.
SOAPElement	addChildElement(java.lang.String localName) ⁷²	Creates a new SOAPElement object initialized with the specified local name and adds the new element to this SOAPElement object.
SOAPElement	addChildElement(java.lang.String localName, java.lang.String prefix) ⁷²	Creates a new SOAPElement object initialized with the specified local name and prefix and adds the new element to this SOAPElement object.
SOAPElement	addChildElement(java.lang.String localName, java.lang.String prefix, java.lang.String uri) ⁷³	Creates a new SOAPElement object initialized with the specified local name, prefix, and URI and adds the new element to this SOAPElement object.
SOAPElement	addNamespaceDeclaration(java.lang.String prefix, java.lang.String uri) ⁷³	Adds a namespace declaration with the specified prefix and URI to this SOAPElement object.
SOAPElement	addTextNode(java.lang.String text) ⁷⁴	Creates a new Text object initialized with the given String and adds it to this SOAPElement object.
javax.xml.namespace.QName	createQName(java.lang.String localName, java.lang.String prefix) ⁷⁴	Creates a QName whose namespace URI is the one associated with the parameter, prefix, in the context of this SOAPElement.
java.util.Iterator<Name>	getAllAttributes() ⁷⁴	Returns an Iterator over all of the attribute Name objects in this SOAPElement object.
java.util.Iterator<javax.xml.namespace.QName>	getAllAttributesAsQNames() ⁷⁵	Returns an Iterator over all of the attributes in this SOAPElement as QName objects.
java.lang.String	getAttributeValue(Name name) ⁷⁵	Returns the value of the attribute with the specified name.
java.lang.String	getAttributeValue(javax.xml.namespace.QName qname) ⁷⁵	Returns the value of the attribute with the specified qname.
java.util.Iterator<Node>	getChildElements() ⁷⁵	Returns an Iterator over all the immediate child Node ₃₅ s of this element.
java.util.Iterator<Node>	getChildElements(Name name) ⁷⁶	Returns an Iterator over all the immediate child Node ₃₅ s of this element with the specified name.
java.util.Iterator<Node>	getChildElements(javax.xml.namespace.QName qname) ⁷⁶	Returns an Iterator over all the immediate child Node ₃₅ s of this element with the specified qname.
Name	getElementName() ⁷⁷	Returns the name of this SOAPElement object.
javax.xml.namespace.QName	getElementQName() ⁷⁷	Returns the qname of this SOAPElement object.

Member Summary	
java.lang.String	getEncodingStyle() ⁷⁷ Returns the encoding style for this SOAPElement object.
java.util.Iterator<String>	getNamespacePrefixes() ⁷⁷ Returns an Iterator over the namespace prefix Strings declared by this element.
java.lang.String	getNamespaceURI(java.lang.String prefix) ⁷⁸ Returns the URI of the namespace that has the given prefix.
java.util.Iterator<String>	getVisibleNamespacePrefixes() ⁷⁸ Returns an Iterator over the namespace prefix Strings visible to this element.
boolean	removeAttribute(Name name) ⁷⁸ Removes the attribute with the specified name.
boolean	removeAttribute(javax.xml.namespace.QName qname) ⁷⁸ Removes the attribute with the specified qname.
void	removeContents() ⁷⁹ Detaches all children of this SOAPElement.
boolean	removeNamespaceDeclaration(java.lang.String prefix) ⁷⁹ Removes the namespace declaration corresponding to the given prefix.
SOAPElement	setElementQName(javax.xml.namespace.QName newName) ⁷⁹ Changes the name of this Element to newName if possible.
void	setEncodingStyle(java.lang.String encodingStyle) ⁷⁹ Sets the encoding style for this SOAPElement object to one specified.

Inherited Member Summary
<p>Fields inherited from interface Node</p> <p>ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE</p>
<p>Methods inherited from interface Element</p> <p>getAttribute(String), getAttributeNS(String, String), getAttributeNode(String), getAttributeNodeNS(String, String), getElementsByTagName(String), getElementsByTagNameNS(String, String), getTagName(), hasAttribute(String), hasAttributeNS(String, String), removeAttribute(String), removeAttributeNS(String, String), removeAttributeNode(Attr), setAttribute(String, String), setAttributeNS(String, String, String), setAttributeNode(Attr), setAttributeNodeNS(Attr)</p>
<p>Methods inherited from interface Node₃₅</p> <p>detachNode() ₃₆, getParentElement() ₃₆, getValue() ₃₆, recycleNode() ₃₇, setParentElement(SOAPElement) ₃₇, setValue(String) ₃₇</p>
<p>Methods inherited from interface Node</p>

Inherited Member Summary

```
appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(),  
getFirstChild(), getLastChild(), getLocalName(), getNamespaceURI(), getNextSibling(),  
getNodeName(), getNodeName(), getNodeValue(), getOwnerDocument(), getParentNode(),  
getPrefix(), getPreviousSibling(), hasAttributes(), hasChildNodes(),  
insertBefore(Node, Node), isSupported(String, String), normalize(),  
removeChild(Node), replaceChild(Node, Node), setNodeValue(String), setPrefix(String)
```

Methods

addAttribute(Name, String)

```
public javax.xml.soap.SOAPElement67 addAttribute(javax.xml.soap.Name32 name,  
    java.lang.String value)  
    throws SOAPException
```

Adds an attribute with the specified name and value to this SOAPElement object.

Parameters:

name - a Name object with the name of the attribute

value - a String giving the value of the attribute

Returns: the SOAPElement object into which the attribute was inserted

Throws:

[SOAPException](#)₈₉ - if there is an error in creating the Attribute, or it is invalid to set an attribute with Name name on this SOAPElement.

addAttribute(QName, String)

```
public javax.xml.soap.SOAPElement67 addAttribute(javax.xml.namespace.QName qname,  
    java.lang.String value)  
    throws SOAPException
```

Adds an attribute with the specified name and value to this SOAPElement object.

Parameters:

qname - a QName object with the name of the attribute

value - a String giving the value of the attribute

Returns: the SOAPElement object into which the attribute was inserted

Throws:

[SOAPException](#)₈₉ - if there is an error in creating the Attribute, or it is invalid to set an attribute with QName qname on this SOAPElement.

Since: SAAJ 1.3

See Also: [addAttribute\(Name, String\)](#) ₇₀

addChildElement(Name)

```
public javax.xml.soap.SOAPElement67 addChildElement(javax.xml.soap.Name32 name)
    throws SOAPException
```

Creates a new SOAPElement object initialized with the given Name object and adds the new element to this SOAPElement object.

This method may be deprecated in a future release of SAAJ in favor of [addChildElement\(QName\)](#) ₇₁

Parameters:

name - a Name object with the XML name for the new element

Returns: the new SOAPElement object that was created

Throws:

[SOAPException](#)₈₉ - if there is an error in creating the SOAPElement object

addChildElement(QName)

```
public javax.xml.soap.SOAPElement67 addChildElement(javax.xml.namespace.QName
    qname)
    throws SOAPException
```

Creates a new SOAPElement object initialized with the given QName object and adds the new element to this SOAPElement object. The *namespace*, *localname* and *prefix* of the new SOAPElement are all taken from the qname argument.

Parameters:

qname - a QName object with the XML name for the new element

Returns: the new SOAPElement object that was created

Throws:

[SOAPException](#)₈₉ - if there is an error in creating the SOAPElement object

Since: SAAJ 1.3

See Also: [addChildElement\(Name\)](#) ₇₁

addChildElement(SOAPElement)

```
public javax.xml.soap.SOAPElement67 addChildElement(javax.xml.soap.SOAPElement67
    element)
    throws SOAPException
```

Add a SOAPElement as a child of this SOAPElement instance. The SOAPElement is expected to be created by a SOAPFactory. Callers should not rely on the element instance being added as is into the XML tree. Implementations could end up copying the content of the SOAPElement passed into an instance of a different SOAPElement implementation. For instance if addChildElement() is called on a SOAPHeader, element will be copied into an instance of a SOAPHeaderElement.

The fragment rooted in element is either added as a whole or not at all, if there was an error.

The fragment rooted in element cannot contain elements named “Envelope”, “Header” or “Body” and in the SOAP namespace. Any namespace prefixes present in the fragment should be fully resolved using appropriate namespace declarations within the fragment itself.

Parameters:

element - the SOAPElement to be added as a new child

Returns: an instance representing the new SOAP element that was actually added to the tree.

Throws:

SOAPException₈₉ - if there was an error in adding this element as a child

addChildElement(String)

```
public javax.xml.soap.SOAPElement67 addChildElement(java.lang.String localName)
    throws SOAPException
```

Creates a new SOAPElement object initialized with the specified local name and adds the new element to this SOAPElement object. The new SOAPElement inherits any in-scope default namespace.

Parameters:

localName - a String giving the local name for the element

Returns: the new SOAPElement object that was created

Throws:

SOAPException₈₉ - if there is an error in creating the SOAPElement object

addChildElement(String, String)

```
public javax.xml.soap.SOAPElement67 addChildElement(java.lang.String localName,
    java.lang.String prefix)
    throws SOAPException
```

Creates a new `SOAPElement` object initialized with the specified local name and prefix and adds the new element to this `SOAPElement` object.

Parameters:

localName - a `String` giving the local name for the new element

prefix - a `String` giving the namespace prefix for the new element

Returns: the new `SOAPElement` object that was created

Throws:

`SOAPException89` - if the prefix is not valid in the context of this `SOAPElement` or if there is an error in creating the `SOAPElement` object

addChildElement(String, String, String)

```
public javax.xml.soap.SOAPElement67 addChildElement(java.lang.String localName,  
    java.lang.String prefix, java.lang.String uri)  
    throws SOAPException
```

Creates a new `SOAPElement` object initialized with the specified local name, prefix, and URI and adds the new element to this `SOAPElement` object.

Parameters:

localName - a `String` giving the local name for the new element

prefix - a `String` giving the namespace prefix for the new element

uri - a `String` giving the URI of the namespace to which the new element belongs

Returns: the new `SOAPElement` object that was created

Throws:

`SOAPException89` - if there is an error in creating the `SOAPElement` object

addNamespaceDeclaration(String, String)

```
public javax.xml.soap.SOAPElement67 addNamespaceDeclaration(java.lang.String  
    prefix, java.lang.String uri)  
    throws SOAPException
```

Adds a namespace declaration with the specified prefix and URI to this `SOAPElement` object.

Parameters:

prefix - a `String` giving the prefix of the namespace

uri - a `String` giving the uri of the namespace

Returns: the `SOAPElement` object into which this namespace declaration was inserted.

Throws:

`SOAPException89` - if there is an error in creating the namespace

addTextNode(String)

```
public javax.xml.soap.SOAPElement67 addTextNode(java.lang.String text)
    throws SOAPException
```

Creates a new `Text` object initialized with the given `String` and adds it to this `SOAPElement` object.

Parameters:

`text` - a `String` object with the textual content to be added

Returns: the `SOAPElement` object into which the new `Text` object was inserted

Throws:

`SOAPException`₈₉ - if there is an error in creating the new `Text` object or if it is not legal to attach it as a child to this `SOAPElement`

createQName(String, String)

```
public javax.xml.namespace.QName createQName(java.lang.String localName,
    java.lang.String prefix)
    throws SOAPException
```

Creates a `QName` whose namespace URI is the one associated with the parameter, `prefix`, in the context of this `SOAPElement`. The remaining elements of the new `QName` are taken directly from the parameters, `localName` and `prefix`.

Parameters:

`localName` - a `String` containing the local part of the name.

`prefix` - a `String` containing the prefix for the name.

Returns: a `QName` with the specified `localName` and `prefix`, and with a namespace that is associated with the `prefix` in the context of this `SOAPElement`. This namespace will be the same as the one that would be returned by `getNamespaceURI(String)`₇₈ if it were given `prefix` as its parameter.

Throws:

`SOAPException`₈₉ - if the `QName` cannot be created.

Since: SAAJ 1.3

getAllAttributes()

```
public java.util.Iterator<Name> getAllAttributes()
```

Returns an `Iterator` over all of the attribute `Name` objects in this `SOAPElement` object. The iterator can be used to get the attribute names, which can then be passed to the method `getAttributeValue` to retrieve the value of each attribute.

Returns: an iterator over the names of the attributes

getAllAttributesAsQNames()

```
public java.util.Iterator<javax.xml.namespace.QName> getAllAttributesAsQNames()
```

Returns an `Iterator` over all of the attributes in this `SOAPElement` as `QName` objects. The iterator can be used to get the attribute `QName`, which can then be passed to the method `getAttributeValue` to retrieve the value of each attribute.

Returns: an iterator over the `QNames` of the attributes

Since: SAAJ 1.3

See Also: [getAttributeValue\(\)](#) ⁷⁴

getAttributeValue(Name)

```
public java.lang.String getAttributeValue(javax.xml.soap.Name32 name)
```

Returns the value of the attribute with the specified name.

Parameters:

name - a `Name` object with the name of the attribute, `Null` if there is no such attribute

Returns: a `String` giving the value of the specified attribute

getAttributeValue(QName)

```
public java.lang.String getAttributeValue(javax.xml.namespace.QName qname)
```

Returns the value of the attribute with the specified `qname`.

Parameters:

qname - a `QName` object with the `qname` of the attribute

Returns: a `String` giving the value of the specified attribute, `Null` if there is no such attribute

Since: SAAJ 1.3

See Also: [getAttributeValue\(Name\)](#) ⁷⁵

getChildElements()

```
public java.util.Iterator<Node35> getChildElements()
```

Returns an `Iterator` over all the immediate child `Node35`s of this element. This includes `javax.xml.soap.Text` objects as well as `SOAPElement` objects.

Calling this method must cause child `Element`, `SOAPElement` and `org.w3c.dom.Text` nodes to be replaced by `SOAPElement`, `SOAPHeaderElement`, `SOAPBodyElement` or `javax.xml.soap.Text` nodes as appropriate for the type of this parent node. As a result the calling application must treat any existing references to these child nodes that have been obtained through DOM APIs as invalid and either discard them or refresh them with the values returned by this `Iterator`. This behavior can be avoided by calling the equivalent DOM APIs. See `javax.xml.soap` ([package-summary.html#package_description](#)) for more details.

Returns: an iterator with the content of this `SOAPElement` object

getChildElements(Name)

```
public java.util.Iterator<Node35> getChildElements(javax.xml.soap.Name32 name)
```

Returns an `Iterator` over all the immediate child [Node₃₅](#)s of this element with the specified name. All of these children will be `SOAPElement` nodes.

Calling this method must cause child `Element`, `SOAPElement` and `org.w3c.dom.Text` nodes to be replaced by `SOAPElement`, `SOAPHeaderElement`, `SOAPBodyElement` or `javax.xml.soap.Text` nodes as appropriate for the type of this parent node. As a result the calling application must treat any existing references to these child nodes that have been obtained through DOM APIs as invalid and either discard them or refresh them with the values returned by this `Iterator`. This behavior can be avoided by calling the equivalent DOM APIs. See `javax.xml.soap` ([package-summary.html#package_description](#)) for more details.

Parameters:

name - a `Name` object with the name of the child elements to be returned

Returns: an `Iterator` object over all the elements in this `SOAPElement` object with the specified name

getChildElements(QName)

```
public java.util.Iterator<Node35> getChildElements(javax.xml.namespace.QName
qname)
```

Returns an `Iterator` over all the immediate child [Node₃₅](#)s of this element with the specified `qname`. All of these children will be `SOAPElement` nodes.

Calling this method must cause child `Element`, `SOAPElement` and `org.w3c.dom.Text` nodes to be replaced by `SOAPElement`, `SOAPHeaderElement`, `SOAPBodyElement` or `javax.xml.soap.Text` nodes as appropriate for the type of this parent node. As a result the calling application must treat any existing references to these child nodes that have been obtained through DOM APIs as invalid and either discard them or refresh them with the values returned by this `Iterator`. This behavior can be avoided by calling the equivalent DOM APIs. See

`javax.xml.soap` ([package-summary.html#package_description](#)) for more details.

Parameters:

`qname` - a `QName` object with the `qname` of the child elements to be returned

Returns: an `Iterator` object over all the elements in this `SOAPElement` object with the specified `qname`

Since: SAAJ 1.3

See Also: [getChildElements\(Name\)](#) ⁷⁶

getElementName()

```
public javax.xml.soap.Name32 getElementName()
```

Returns the name of this `SOAPElement` object.

Returns: a `Name` object with the name of this `SOAPElement` object

getElementQName()

```
public javax.xml.namespace.QName getElementQName()
```

Returns the `qname` of this `SOAPElement` object.

Returns: a `QName` object with the `qname` of this `SOAPElement` object

Since: SAAJ 1.3

See Also: [getElementName\(\)](#) ⁷⁷

getEncodingStyle()

```
public java.lang.String getEncodingStyle()
```

Returns the encoding style for this `SOAPElement` object.

Returns: a `String` giving the encoding style

See Also: [setEncodingStyle\(String\)](#) ⁷⁹

getNamespacePrefixes()

```
public java.util.Iterator<String> getNamespacePrefixes()
```

Returns an `Iterator` over the namespace prefix `Strings` declared by this element. The prefixes returned by this iterator can be passed to the method `getNamespaceURI` to retrieve the `URI` of each namespace.

Returns: an iterator over the namespace prefixes in this SOAPElement object

getNamespaceURI(String)

```
public java.lang.String getNamespaceURI(java.lang.String prefix)
```

Returns the URI of the namespace that has the given prefix.

Parameters:

prefix - a String giving the prefix of the namespace for which to search

Returns: a String with the uri of the namespace that has the given prefix

getVisibleNamespacePrefixes()

```
public java.util.Iterator<String> getVisibleNamespacePrefixes()
```

Returns an Iterator over the namespace prefix Strings visible to this element. The prefixes returned by this iterator can be passed to the method `getNamespaceURI` to retrieve the URI of each namespace.

Returns: an iterator over the namespace prefixes are within scope of this SOAPElement object

Since: SAAJ 1.2

removeAttribute(Name)

```
public boolean removeAttribute(javax.xml.soap.Name32 name)
```

Removes the attribute with the specified name.

Parameters:

name - the Name object with the name of the attribute to be removed

Returns: true if the attribute was removed successfully; false if it was not

removeAttribute(QName)

```
public boolean removeAttribute(javax.xml.namespace.QName qname)
```

Removes the attribute with the specified qname.

Parameters:

qname - the QName object with the qname of the attribute to be removed

Returns: true if the attribute was removed successfully; false if it was not

Since: SAAJ 1.3

See Also: [removeAttribute\(Name\)](#) ₇₈

removeContents()

```
public void removeContents()
```

Detaches all children of this SOAPElement.

This method is useful for rolling back the construction of partially completed SOAPHeaders and SOAPBodys in preparation for sending a fault when an error condition is detected. It is also useful for recycling portions of a document within a SOAP message.

Since: SAAJ 1.2

removeNamespaceDeclaration(String)

```
public boolean removeNamespaceDeclaration(java.lang.String prefix)
```

Removes the namespace declaration corresponding to the given prefix.

Parameters:

prefix - a String giving the prefix for which to search

Returns: true if the namespace declaration was removed successfully; false if it was not

setElementQName(QName)

```
public javax.xml.soap.SOAPElement67 setElementQName(javax.xml.namespace.QName  
newName)  
throws SOAPException
```

Changes the name of this Element to newName if possible. SOAP Defined elements such as SOAPEnvelope, SOAPHeader, SOAPBody etc. cannot have their names changed using this method. Any attempt to do so will result in a SOAPException being thrown.

Callers should not rely on the element instance being renamed as is. Implementations could end up copying the content of the SOAPElement to a renamed instance.

Parameters:

newName - the new name for the Element.

Returns: The renamed Node

Throws:

SOAPException₈₉ - if changing the name of this Element is not allowed.

Since: SAAJ 1.3

setEncodingStyle(String)

```
public void setEncodingStyle(java.lang.String encodingStyle)  
throws SOAPException
```

Sets the encoding style for this `SOAPElement` object to one specified.

Parameters:

`encodingStyle` - a `String` giving the encoding style

Throws:

`java.lang.IllegalArgumentException` - if there was a problem in the encoding style being set.

[SOAPException₈₉](#) - if setting the `encodingStyle` is invalid for this `SOAPElement`.

See Also: [getEncodingStyle\(\)₇₇](#)

2.18 SOAPElementFactory

Declaration

```
public class SOAPElementFactory
    java.lang.Object
    |
    +--javax.xml.soap.SOAPElementFactory
```

Description

Deprecated. - Use `javax.xml.soap.SOAPFactory` for creating `SOAPElements`.

`SOAPElementFactory` is a factory for XML fragments that will eventually end up in the SOAP part. These fragments can be inserted as children of the `SOAPHeader` or `SOAPBody` or `SOAPEnvelope`.

Elements created using this factory do not have the properties of an element that lives inside a SOAP header document. These elements are copied into the XML document tree when they are inserted.

See Also: [SOAPFactory₉₃](#)

Member Summary		
Methods		
SOAPElement	<code>create(Name name)₈₂</code>	Create a <code>SOAPElement</code> object initialized with the given <code>Name</code> object.
SOAPElement	<code>create(java.lang.String localName)₈₂</code>	Create a <code>SOAPElement</code> object initialized with the given local name.
SOAPElement	<code>create(java.lang.String localName, java.lang.String prefix, java.lang.String uri)₈₃</code>	Create a new <code>SOAPElement</code> object with the given local name, prefix and uri.
static SOAPElementFactory	<code>newInstance()₈₃</code>	Creates a new instance of <code>SOAPElementFactory</code> .

Inherited Member Summary

Methods inherited from class `Object`

`clone()`, `equals(Object)`, `finalize()`, `getClass()`, `hashCode()`, `notify()`, `notifyAll()`, `toString()`, `wait(long, int)`, `wait(long, int)`, `wait(long, int)`

Methods

`create(Name)`

```
public javax.xml.soap.SOAPElement67 create(javax.xml.soap.Name32 name)
    throws SOAPException
```

Deprecated. Use `javax.xml.soap.SOAPFactory.createElement(javax.xml.soap.Name)` instead

Create a `SOAPElement` object initialized with the given `Name` object.

Parameters:

`name` - a `Name` object with the XML name for the new element

Returns: the new `SOAPElement` object that was created

Throws:

`SOAPException89` - if there is an error in creating the `SOAPElement` object

See Also: `SOAPFactory.createElement(Name)95`

`create(String)`

```
public javax.xml.soap.SOAPElement67 create(java.lang.String localName)
    throws SOAPException
```

Deprecated. Use `javax.xml.soap.SOAPFactory.createElement(String localName)` instead

Create a `SOAPElement` object initialized with the given local name.

Parameters:

`localName` - a `String` giving the local name for the new element

Returns: the new `SOAPElement` object that was created

Throws:

`SOAPException89` - if there is an error in creating the `SOAPElement` object

See Also: `SOAPFactory.createElement(String)96`

create(String, String, String)

```
public javax.xml.soap.SOAPElement67 create(java.lang.String localName,  
        java.lang.String prefix, java.lang.String uri)  
        throws SOAPException
```

Deprecated. Use `javax.xml.soap.SOAPFactory.createElement(String localName, String prefix, String uri)` instead

Create a new `SOAPElement` object with the given local name, prefix and uri.

Returns: the new `SOAPElement` object that was created

Parameters:

`localName` - a `String` giving the local name for the new element

`prefix` - the prefix for this `SOAPElement`

`uri` - a `String` giving the URI of the namespace to which the new element belongs

Throws:

`SOAPException89` - if there is an error in creating the `SOAPElement` object

See Also: `SOAPFactory.createElement(String, String, String)96`

newInstance()

```
public static javax.xml.soap.SOAPElementFactory81 newInstance()  
        throws SOAPException
```

Creates a new instance of `SOAPElementFactory`.

Returns: a new instance of a `SOAPElementFactory`

Throws:

`SOAPException89` - if there was an error creating the default `SOAPElementFactory`

2.19 SOAPEnvelope

Declaration

```
public interface SOAPEnvelope extends SOAPElement67
```

All Superinterfaces: `org.w3c.dom.Element`, `org.w3c.dom.Node`, `Node35`, `SOAPElement67`

Description

The container for the SOAPHeader and SOAPBody portions of a SOAPPart object. By default, a SOAPMessage object is created with a SOAPPart object that has a SOAPEnvelope object. The SOAPEnvelope object by default has an empty SOAPBody object and an empty SOAPHeader object. The SOAPBody object is required, and the SOAPHeader object, though optional, is used in the majority of cases. If the SOAPHeader object is not needed, it can be deleted, which is shown later.

A client can access the SOAPHeader and SOAPBody objects by calling the methods `SOAPEnvelope.getHeader()` and `SOAPEnvelope.getBody()`. The following lines of code use these two methods after starting with the SOAPMessage object *message* to get the SOAPPart object *sp*, which is then used to get the SOAPEnvelope object *se*.

```
SOAPPart sp = message.getSOAPPart();
SOAPEnvelope se = sp.getEnvelope();
SOAPHeader sh = se.getHeader();
SOAPBody sb = se.getBody();
```

It is possible to change the body or header of a SOAPEnvelope object by retrieving the current one, deleting it, and then adding a new body or header. The `javax.xml.soap.Node` method `deleteNode` deletes the XML element (node) on which it is called. For example, the following line of code deletes the SOAPBody object that is retrieved by the method `getBody()`.

```
se.getBody().detachNode();
```

To create a SOAPHeader object to replace the one that was removed, a client uses the method `SOAPEnvelope.addHeader()`, which creates a new header and adds it to the SOAPEnvelope object. Similarly, the method `addBody()` creates a new SOAPBody object and adds it to the SOAPEnvelope object. The following code fragment retrieves the current header, removes it, and adds a new one. Then it retrieves the current body, removes it, and adds a new one.

```

SOAPPart sp = message.getSOAPPart();
SOAPEnvelope se = sp.getEnvelope();
se.getHeader().detachNode();
SOAPHeader sh = se.addHeader();
se.getBody().detachNode();
SOAPBody sb = se.addBody();

```

It is an error to add a SOAPBody or SOAPHeader object if one already exists.

The SOAPEnvelope interface provides three methods for creating Name objects. One method creates Name objects with a local name, a namespace prefix, and a namespace URI. The second method creates Name objects with a local name and a namespace prefix, and the third creates Name objects with just a local name. The following line of code, in which *se* is a SOAPEnvelope object, creates a new Name object with all three.

```

Name name = se.createName("GetLastTradePrice", "WOMBAT",
                          "http://www.wombat.org/trader");

```

Member Summary

Methods

SOAPBody	addBody() ⁸⁶	Creates a SOAPBody object and sets it as the SOAPBody object for this SOAPEnvelope object.
SOAPHeader	addHeader() ⁸⁷	Creates a SOAPHeader object and sets it as the SOAPHeader object for this SOAPEnvelope object.
Name	createName(java.lang.String localName) ⁸⁷	Creates a new Name object initialized with the given local name.
Name	createName(java.lang.String localName, java.lang.String prefix, java.lang.String uri) ⁸⁷	Creates a new Name object initialized with the given local name, namespace prefix, and namespace URI.
SOAPBody	getBody() ⁸⁸	Returns the SOAPBody object associated with this SOAPEnvelope object.
SOAPHeader	getHeader() ⁸⁸	Returns the SOAPHeader object for this SOAPEnvelope object.

Inherited Member Summary

Fields inherited from interface Node

Inherited Member Summary

ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE

Methods inherited from interface **Element**

getAttribute(String), getAttributeNS(String, String), getAttributeNode(String), getAttributeNodeNS(String, String), getElementsByTagName(String), getElementsByTagNameNS(String, String), getTagName(), hasAttribute(String), hasAttributeNS(String, String), removeAttribute(String), removeAttributeNS(String, String), removeAttributeNode(Attr), setAttribute(String, String), setAttributeNS(String, String, String), setAttributeNode(Attr), setAttributeNodeNS(Attr)

Methods inherited from interface **Node₃₅**

detachNode()₃₆, getParentElement()₃₆, getValue()₃₆, recycleNode()₃₇, setParentElement(SOAPElement)₃₇, setValue(String)₃₇

Methods inherited from interface **Node**

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getLocalName(), getNamespaceURI(), getNextSibling(), getNodeName(), getNodeType(), getNodeValue(), getOwnerDocument(), getParentNode(), getPrefix(), getPreviousSibling(), hasAttributes(), hasChildNodes(), insertBefore(Node, Node), isSupported(String, String), normalize(), removeChild(Node), replaceChild(Node, Node), setNodeValue(String), setPrefix(String)

Methods inherited from interface **SOAPElement₆₇**

addAttribute(Name, String)₇₀, addChildElement(SOAPElement)₇₂, addChildElement(SOAPElement)₇₂, addChildElement(SOAPElement)₇₂, addChildElement(SOAPElement)₇₂, addNamespaceDeclaration(String, String)₇₃, addTextNode(String)₇₄, getAllAttributes()₇₄, getAttributeValue(Name)₇₅, getChildElements(Name)₇₆, getChildElements(Name)₇₆, getElementName()₇₇, getEncodingStyle()₇₇, getNamespacePrefixes()₇₇, getNamespaceURI(String)₇₈, getVisibleNamespacePrefixes()₇₈, removeAttribute(Name)₇₈, removeContents()₇₉, removeNamespaceDeclaration(String)₇₉, setEncodingStyle(String)₇₉

Methods

addBody()

```
public javax.xml.soap.SOAPBody46 addBody()
    throws SOAPException
```

Creates a SOAPBody object and sets it as the SOAPBody object for this SOAPEnvelope object. It is illegal to add a body when the envelope already contains a body. Therefore, this method should be called only after the existing body has been removed.

Returns: the new SOAPBody object

Throws:

`SOAPException89` - if this SOAPEnvelope object already contains a valid SOAPBody object

addHeader()

```
public javax.xml.soap.SOAPHeader115 addHeader()  
    throws SOAPException
```

Creates a SOAPHeader object and sets it as the SOAPHeader object for this SOAPEnvelope object.

It is illegal to add a header when the envelope already contains a header. Therefore, this method should be called only after the existing header has been removed.

Returns: the new SOAPHeader object

Throws:

`SOAPException89` - if this SOAPEnvelope object already contains a valid SOAPHeader object

createName(String)

```
public javax.xml.soap.Name32 createName(java.lang.String localName)  
    throws SOAPException
```

Creates a new Name object initialized with the given local name.

This factory method creates Name objects for use in the SOAP/XML document.

Parameters:

`localName` - a String giving the local name

Returns: a Name object initialized with the given local name

Throws:

`SOAPException89` - if there is a SOAP error

createName(String, String, String)

```
public javax.xml.soap.Name32 createName(java.lang.String localName,  
    java.lang.String prefix, java.lang.String uri)  
    throws SOAPException
```

Creates a new Name object initialized with the given local name, namespace prefix, and namespace URI.

This factory method creates Name objects for use in the SOAP/XML document.

Parameters:

localName - a String giving the local name

prefix - a String giving the prefix of the namespace

uri - a String giving the URI of the namespace

Returns: a Name object initialized with the given local name, namespace prefix, and namespace URI

Throws:

`SOAPException89` - if there is a SOAP error

getBody()

```
public javax.xml.soap.SOAPBody46 getBody()  
    throws SOAPException
```

Returns the SOAPBody object associated with this SOAPEnvelope object.

A new SOAPMessage object is by default created with a SOAPEnvelope object that contains an empty SOAPBody object. As a result, the method getBody will always return a SOAPBody object unless the body has been removed and a new one has not been added.

Returns: the SOAPBody object for this SOAPEnvelope object or null if there is none

Throws:

`SOAPException89` - if there is a problem obtaining the SOAPBody object

getHeader()

```
public javax.xml.soap.SOAPHeader115 getHeader()  
    throws SOAPException
```

Returns the SOAPHeader object for this SOAPEnvelope object.

A new SOAPMessage object is by default created with a SOAPEnvelope object that contains an empty SOAPHeader object. As a result, the method getHeader will always return a SOAPHeader object unless the header has been removed and a new one has not been added.

Returns: the SOAPHeader object or null if there is none

Throws:

`SOAPException89` - if there is a problem obtaining the SOAPHeader object

2.20 SOAPException

Declaration

public class **SOAPException** extends java.lang.Exception

```
java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--javax.xml.soap.SOAPException
```

All Implemented Interfaces: java.io.Serializable

Description

An exception that signals that a SOAP exception has occurred. A `SOAPException` object may contain a `String` that gives the reason for the exception, an embedded `Throwable` object, or both. This class provides methods for retrieving reason messages and for retrieving the embedded `Throwable` object.

Typical reasons for throwing a `SOAPException` object are problems such as difficulty setting a header, not being able to send a message, and not being able to get a connection with the provider. Reasons for embedding a `Throwable` object include problems such as input/output errors or a parsing problem, such as an error in parsing a header.

Member Summary

Constructors

`SOAPException()` [90](#)

Constructs a `SOAPException` object with no reason or embedded `Throwable` object.

`SOAPException(java.lang.String reason)` [90](#)

Constructs a `SOAPException` object with the given `String` as the reason for the exception being thrown.

`SOAPException(java.lang.String reason, java.lang.Throwable cause)` [91](#)

Constructs a `SOAPException` object with the given `String` as the reason for the exception being thrown and the given `Throwable` object as an embedded exception.

Member Summary	
	<code>SOAPException(java.lang.Throwable cause)</code> ⁹¹ Constructs a SOAPException object initialized with the given Throwable object.
Methods	
<code>java.lang.Throwable</code>	<code>getCause()</code> ⁹¹ Returns the Throwable object embedded in this SOAPException if there is one.
<code>java.lang.String</code>	<code>getMessage()</code> ⁹¹ Returns the detail message for this SOAPException object.
<code>java.lang.Throwable</code>	<code>initCause(java.lang.Throwable cause)</code> ⁹² Initializes the cause field of this SOAPException object with the given Throwable object.

Inherited Member Summary
Methods inherited from class Object <code>clone()</code> , <code>equals(Object)</code> , <code>finalize()</code> , <code>getClass()</code> , <code>hashCode()</code> , <code>notify()</code> , <code>notifyAll()</code> , <code>wait(long, int)</code> , <code>wait(long, int)</code> , <code>wait(long, int)</code>
Methods inherited from class Throwable <code>fillInStackTrace()</code> , <code>getLocalizedMessage()</code> , <code>getStackTrace()</code> , <code>printStackTrace(PrintWriter)</code> , <code>printStackTrace(PrintWriter)</code> , <code>printStackTrace(PrintWriter)</code> , <code>setStackTrace(StackTraceElement[])</code> , <code>toString()</code>

Constructors

SOAPException()

```
public SOAPException()
```

Constructs a SOAPException object with no reason or embedded Throwable object.

SOAPException(String)

```
public SOAPException(java.lang.String reason)
```

Constructs a SOAPException object with the given String as the reason for the exception being thrown.

Parameters:

`reason` - a description of what caused the exception

SOAPException(String, Throwable)

```
public SOAPException(java.lang.String reason, java.lang.Throwable cause)
```

Constructs a SOAPException object with the given String as the reason for the exception being thrown and the given Throwable object as an embedded exception.

Parameters:

reason - a description of what caused the exception

cause - a Throwable object that is to be embedded in this SOAPException object

SOAPException(Throwable)

```
public SOAPException(java.lang.Throwable cause)
```

Constructs a SOAPException object initialized with the given Throwable object.

Parameters:

cause - a Throwable object that is to be embedded in this SOAPException object

Methods

getCause()

```
public java.lang.Throwable getCause()
```

Returns the Throwable object embedded in this SOAPException if there is one. Otherwise, this method returns null.

Overrides: getCause in class Throwable

Returns: the embedded Throwable object or null if there is none

getMessage()

```
public java.lang.String getMessage()
```

Returns the detail message for this SOAPException object.

If there is an embedded Throwable object, and if the SOAPException object has no detail message of its own, this method will return the detail message from the embedded Throwable object.

Overrides: getMessage in class Throwable

Returns: the error or warning message for this SOAPException or, if it has none, the message of the embedded Throwable object, if there is one

initCause(Throwable)

```
public java.lang.Throwable initCause(java.lang.Throwable cause)
```

Initializes the cause field of this `SOAPException` object with the given `Throwable` object.

This method can be called at most once. It is generally called from within the constructor or immediately after the constructor has returned a new `SOAPException` object. If this `SOAPException` object was created with the constructor `SOAPException(Throwable)`⁹¹ or `SOAPException(String, Throwable)`⁹¹, meaning that its cause field already has a value, this method cannot be called even once.

Overrides: `initCause` in class `Throwable`

Parameters:

`cause` - the `Throwable` object that caused this `SOAPException` object to be thrown. The value of this parameter is saved for later retrieval by the `getCause()`⁹¹ method. A null value is permitted and indicates that the cause is nonexistent or unknown.

Returns: a reference to this `SOAPException` instance

Throws:

`java.lang.IllegalArgumentException` - if `cause` is this `Throwable` object. (A `Throwable` object cannot be its own cause.)

`java.lang.IllegalStateException` - if the cause for this `SOAPException` object has already been initialized

2.21 SOAPFactory

Declaration

```
public abstract class SOAPFactory
```

```
java.lang.Object
```

```
|
+-- javax.xml.soap.SOAPFactory
```

Description

SOAPFactory is a factory for creating various objects that exist in the SOAP XML tree.

SOAPFactory can be used to create XML fragments that will eventually end up in the SOAP part.

These fragments can be inserted as children of the [SOAPHeaderElement](#)₁₂₃ or [SOAPBodyElement](#)₅₄ or [SOAPEnvelope](#)₈₄ or other [SOAPElement](#)₆₇ objects. SOAPFactory also has methods to create `javax.xml.soap.Detail` objects as well as `javax.xml.soap.Name` objects.

Member Summary

Constructors

[SOAPFactory\(\)](#)₉₄

Methods

abstract Detail [createDetail\(\)](#)₉₄
Creates a new Detail object which serves as a container for DetailEntry objects.

SOAPElement [createElement\(org.w3c.dom.Element domElement\)](#)₉₅
Creates a SOAPElement object from an existing DOM Element.

abstract SOAPElement [createElement\(Name name\)](#)₉₅
Create a SOAPElement object initialized with the given Name object.

SOAPElement [createElement\(javax.xml.namespace.QName qname\)](#)₉₆
Creates a SOAPElement object initialized with the given QName object.

abstract SOAPElement [createElement\(java.lang.String localName\)](#)₉₆
Create a SOAPElement object initialized with the given local name.

abstract SOAPElement [createElement\(java.lang.String localName, java.lang.String prefix, java.lang.String uri\)](#)₉₆
Create a new SOAPElement object with the given local name, prefix and uri.

Member Summary

abstract SOAPFault	<code>createFault()</code> ⁹⁷	Creates a new default SOAPFault object
abstract SOAPFault	<code>createFault(java.lang.String reasonText, javax.xml.namespace.QName faultCode)</code> ⁹⁷	Creates a new SOAPFault object initialized with the given reasonText and faultCode
abstract Name	<code>createName(java.lang.String localName)</code> ⁹⁸	Creates a new Name object initialized with the given local name.
abstract Name	<code>createName(java.lang.String localName, java.lang.String prefix, java.lang.String uri)</code> ⁹⁸	Creates a new Name object initialized with the given local name, namespace prefix, and namespace URI.
static SOAPFactory	<code>newInstance()</code> ⁹⁸	Creates a new SOAPFactory object that is an instance of the default implementation (SOAP 1.1). This method uses the following ordered lookup procedure to determine the SOAPFactory implementation class to load: Use the javax.xml.soap.SOAPFactory system property.
static SOAPFactory	<code>newInstance(java.lang.String protocol)</code> ⁰	Creates a new SOAPFactory object that is an instance of the specified implementation, this method uses the SAAJMetaFactory ₃₉ to locate the implementation class and create the SOAPFactory instance.

Inherited Member Summary

Methods inherited from class Object

`clone()`, `equals(Object)`, `finalize()`, `getClass()`, `hashCode()`, `notify()`, `notifyAll()`, `toString()`, `wait(long, int)`, `wait(long, int)`, `wait(long, int)`

Constructors

SOAPFactory()

```
public SOAPFactory()
```

Methods

createDetail()

```
public abstract javax.xml.soap.Detail17 createDetail()  
    throws SOAPException
```

Creates a new `Detail` object which serves as a container for `DetailEntry` objects.

This factory method creates `Detail` objects for use in situations where it is not practical to use the `SOAPFault` abstraction.

Returns: a `Detail` object

Throws:

`SOAPException`₈₉ - if there is a SOAP error

`java.lang.UnsupportedOperationException` - if the protocol specified for the `SOAPFactory` was `DYNAMIC_SOAP_PROTOCOL`₆₂

createElement(Element)

```
public javax.xml.soap.SOAPElement67 createElement(org.w3c.dom.Element domElement)
    throws SOAPException
```

Creates a `SOAPElement` object from an existing DOM `Element`. If the DOM `Element` that is passed in as an argument is already a `SOAPElement` then this method must return it unmodified without any further work. Otherwise, a new `SOAPElement` is created and a deep copy is made of the `domElement` argument. The concrete type of the return value will depend on the name of the `domElement` argument. If any part of the tree rooted in `domElement` violates SOAP rules, a `SOAPException` will be thrown.

Parameters:

`domElement` - the `Element` to be copied.

Returns: a new `SOAPElement` that is a copy of `domElement`.

Throws:

`SOAPException`₈₉ - if there is an error in creating the `SOAPElement` object

Since: SAAJ 1.3

createElement(Name)

```
public abstract javax.xml.soap.SOAPElement67 createElement(javax.xml.soap.Name32
    name)
    throws SOAPException
```

Create a `SOAPElement` object initialized with the given `Name` object. The concrete type of the return value will depend on the name given to the new `SOAPElement`. For instance, a new `SOAPElement` with the name “`{http://www.w3.org/2003/05/soap-envelope}Envelope`” would cause a `SOAPEnvelope` that supports SOAP 1.2 behavior to be created.

Parameters:

`name` - a `Name` object with the XML name for the new element

Returns: the new `SOAPElement` object that was created

Throws:

`SOAPException`₈₉ - if there is an error in creating the `SOAPElement` object

See Also: `createElement(QName)`₉₆

createElement(QName)

```
public javax.xml.soap.SOAPElement67 createElement(javax.xml.namespace.QName qname)
    throws SOAPException
```

Creates a `SOAPElement` object initialized with the given `QName` object. The concrete type of the return value will depend on the name given to the new `SOAPElement`. For instance, a new `SOAPElement` with the name “{http://www.w3.org/2003/05/soap-envelope}Envelope” would cause a `SOAPEnvelope` that supports SOAP 1.2 behavior to be created.

Parameters:

`qname` - a `QName` object with the XML name for the new element

Returns: the new `SOAPElement` object that was created

Throws:

`SOAPException`₈₉ - if there is an error in creating the `SOAPElement` object

Since: SAAJ 1.3

See Also: `createElement(Name)`₉₅

createElement(String)

```
public abstract javax.xml.soap.SOAPElement67 createElement(java.lang.String
    localName)
    throws SOAPException
```

Create a `SOAPElement` object initialized with the given local name.

Parameters:

`localName` - a `String` giving the local name for the new element

Returns: the new `SOAPElement` object that was created

Throws:

`SOAPException`₈₉ - if there is an error in creating the `SOAPElement` object

createElement(String, String, String)

```
public abstract javax.xml.soap.SOAPElement67 createElement(java.lang.String
    localName, java.lang.String prefix, java.lang.String uri)
    throws SOAPException
```

Create a new `SOAPElement` object with the given local name, prefix and uri. The concrete type of the return value will depend on the name given to the new `SOAPElement`. For instance, a new `SOAPElement` with the name “`{http://www.w3.org/2003/05/soap-envelope}Envelope`” would cause a `SOAPEnvelope` that supports SOAP 1.2 behavior to be created.

Parameters:

`localName` - a `String` giving the local name for the new element

`prefix` - the prefix for this `SOAPElement`

`uri` - a `String` giving the URI of the namespace to which the new element belongs

Returns: the new `SOAPElement` object that was created

Throws:

`SOAPException89` - if there is an error in creating the `SOAPElement` object

createFault()

```
public abstract javax.xml.soap.SOAPFault100 createFault()  
    throws SOAPException
```

Creates a new default `SOAPFault` object

Returns: a `SOAPFault` object

Throws:

`SOAPException89` - if there is a SOAP error

Since: SAAJ 1.3

createFault(String, QName)

```
public abstract javax.xml.soap.SOAPFault100 createFault(java.lang.String  
    reasonText, javax.xml.namespace.QName faultCode)  
    throws SOAPException
```

Creates a new `SOAPFault` object initialized with the given `reasonText` and `faultCode`

Parameters:

`reasonText` - the `ReasonText/FaultString` for the fault

`faultCode` - the `FaultCode` for the fault

Returns: a `SOAPFault` object

Throws:

`SOAPException89` - if there is a SOAP error

Since: SAAJ 1.3

createName(String)

```
public abstract javax.xml.soap.Name32 createName(java.lang.String localName)
    throws SOAPException
```

Creates a new Name object initialized with the given local name.

This factory method creates Name objects for use in situations where it is not practical to use the SOAPEnvelope abstraction.

Parameters:

localName - a String giving the local name

Returns: a Name object initialized with the given local name

Throws:

SOAPException₈₉ - if there is a SOAP error

createName(String, String, String)

```
public abstract javax.xml.soap.Name32 createName(java.lang.String localName,
    java.lang.String prefix, java.lang.String uri)
    throws SOAPException
```

Creates a new Name object initialized with the given local name, namespace prefix, and namespace URI.

This factory method creates Name objects for use in situations where it is not practical to use the SOAPEnvelope abstraction.

Parameters:

localName - a String giving the local name

prefix - a String giving the prefix of the namespace

uri - a String giving the URI of the namespace

Returns: a Name object initialized with the given local name, namespace prefix, and namespace URI

Throws:

SOAPException₈₉ - if there is a SOAP error

newInstance()

```
public static javax.xml.soap.SOAFFactory93 newInstance()
    throws SOAPException
```

Creates a new `SOAPFactory` object that is an instance of the default implementation (SOAP 1.1). This method uses the lookup procedure specified in `javax.xml.soap` to locate and load the `SOAPFactory` class.

Returns: a new instance of a `SOAPFactory`

Throws:

`SOAPException89` - if there was an error creating the default `SOAPFactory`

See Also: `SAAJMetaFactory39`

newInstance(String)

```
public static javax.xml.soap.SOAPFactory93 newInstance(java.lang.String protocol)
    throws SOAPException
```

Creates a new `SOAPFactory` object that is an instance of the specified implementation, this method uses the `SAAJMetaFactory39` to locate the implementation class and create the `SOAPFactory` instance.

Parameters:

`protocol` - a string constant representing the protocol of the specified SOAP factory implementation. May be either `DYNAMIC_SOAP_PROTOCOL`, `DEFAULT_SOAP_PROTOCOL` (which is the same as) `SOAP_1_1_PROTOCOL`, or `SOAP_1_2_PROTOCOL`.

Returns: a new instance of a `SOAPFactory`

Throws:

`SOAPException89` - if there was an error creating the specified `SOAPFactory`

See Also: `SAAJMetaFactory39`

Since: SAAJ 1.3

2.22 SOAPFault

Declaration

```
public interface SOAPFault extends SOAPBodyElement54
```

All Superinterfaces: `org.w3c.dom.Element`, `org.w3c.dom.Node`, [Node₃₅](#),
[SOAPBodyElement₅₄](#), [SOAPElement₆₇](#)

Description

An element in the SOAPBody object that contains error and/or status information. This information may relate to errors in the SOAPMessage object or to problems that are not related to the content in the message itself. Problems not related to the message itself are generally errors in processing, such as the inability to communicate with an upstream server.

Depending on the protocol specified while creating the MessageFactory instance, a SOAPFault has sub-elements as defined in the SOAP 1.1/SOAP 1.2 specification.

Member Summary	
Methods	
Detail	<code>addDetail()₁₀₃</code> Creates an optional Detail object and sets it as the Detail object for this SOAPFault object.
void	<code>addFaultReasonText(java.lang.String text, java.util.Locale locale)₁₀₃</code> Appends or replaces a Reason Text item containing the specified text message and an <i>xml:lang</i> derived from locale.
void	<code>appendFaultSubcode(javax.xml.namespace.QName subcode)₁₀₃</code> Adds a Subcode to the end of the sequence of Subcodes contained by this SOAPFault.
Detail	<code>getDetail()₁₀₄</code> Returns the optional detail element for this SOAPFault object.
java.lang.String	<code>getFaultActor()₁₀₄</code> Gets the fault actor for this SOAPFault object.
java.lang.String	<code>getFaultCode()₁₀₄</code> Gets the fault code for this SOAPFault object.

Member Summary

Name	<code>getFaultCodeAsName()</code> ¹⁰⁴	Gets the mandatory SOAP 1.1 fault code for this SOAPFault object as a SAAJ Name object.
<code>javax.xml.namespace.QName</code>	<code>getFaultCodeAsQName()</code> ¹⁰⁵	Gets the fault code for this SOAPFault object as a QName object.
<code>java.lang.String</code>	<code>getNode()</code> ¹⁰⁵	Returns the optional Node element value for this SOAPFault object.
<code>java.util.Iterator<java.util.Locale></code>	<code>getFaultReasonLocales()</code> ¹⁰⁵	Returns an Iterator over a distinct sequence of Locales for which there are associated Reason Text items.
<code>java.lang.String</code>	<code>getFaultReasonText(java.util.Locale locale)</code> ¹⁰⁶	Returns the Reason Text associated with the given Locale.
<code>java.util.Iterator<String></code>	<code>getFaultReasonTexts()</code> ¹⁰⁶	Returns an Iterator over a sequence of String objects containing all of the Reason Text items for this SOAPFault.
<code>java.lang.String</code>	<code>getFaultRole()</code> ¹⁰⁷	Returns the optional Role element value for this SOAPFault object.
<code>java.lang.String</code>	<code>getFaultString()</code> ¹⁰⁷	Gets the fault string for this SOAPFault object.
<code>java.util.Locale</code>	<code>getFaultStringLocale()</code> ¹⁰⁷	Gets the locale of the fault string for this SOAPFault object.
<code>java.util.Iterator<javax.xml.namespace.QName></code>	<code>getFaultSubcodes()</code> ¹⁰⁸	Gets the Subcodes for this SOAPFault as an iterator over QNames.
<code>boolean</code>	<code>hasDetail()</code> ¹⁰⁸	Returns true if this SOAPFault has a Detail subelement and false otherwise.
<code>void</code>	<code>removeAllFaultSubcodes()</code> ¹⁰⁸	Removes any Subcodes that may be contained by this SOAPFault.
<code>void</code>	<code>setFaultActor(java.lang.String faultActor)</code> ¹⁰⁸	Sets this SOAPFault object with the given fault actor.
<code>void</code>	<code>setFaultCode(Name faultCodeQName)</code> ¹⁰⁹	Sets this SOAPFault object with the given fault code.
<code>void</code>	<code>setFaultCode(java.lang.String faultCode)</code> ¹¹⁰	Sets this SOAPFault object with the give fault code.
<code>void</code>	<code>setFaultRole(java.lang.String uri)</code> ¹¹¹	Creates or replaces any existing Role element value for this SOAPFault object.
<code>void</code>	<code>setFaultString(java.lang.String faultString)</code> ¹¹¹	Sets the fault string for this SOAPFault object to the given string.
<code>void</code>	<code>setFaultString(java.lang.String faultString, java.util.Locale locale)</code> ¹¹¹	Sets the fault string for this SOAPFault object to the given string and localized to the given locale.

Inherited Member Summary

Fields inherited from interface **Node**

ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE

Methods inherited from interface **Element**

getAttribute(String), getAttributeNS(String, String), getAttributeNode(String), getAttributeNodeNS(String, String), getElementsByTagName(String), getElementsByTagNameNS(String, String), getTagName(), hasAttribute(String), hasAttributeNS(String, String), removeAttribute(String), removeAttributeNS(String, String), removeAttributeNode(Attr), setAttribute(String, String), setAttributeNS(String, String, String), setAttributeNode(Attr), setAttributeNodeNS(Attr)

Methods inherited from interface **Node₃₅**

detachNode()₃₆, getParentElement()₃₆, getValue()₃₆, recycleNode()₃₇, setParentElement(SOAPElement)₃₇, setValue(String)₃₇

Methods inherited from interface **Node**

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getLocalName(), getNamespaceURI(), getNextSibling(), getNodeName(), getNodeType(), getNodeValue(), getOwnerDocument(), getParentNode(), getPrefix(), getPreviousSibling(), hasAttributes(), hasChildNodes(), insertBefore(Node, Node), isSupported(String, String), normalize(), removeChild(Node), replaceChild(Node, Node), setNodeValue(String), setPrefix(String)

Methods inherited from interface **SOAPElement₆₇**

addAttribute(Name, String)₇₀, addChildElement(SOAPElement)₇₂, addChildElement(SOAPElement)₇₂, addChildElement(SOAPElement)₇₂, addChildElement(SOAPElement)₇₂, addNamespaceDeclaration(String, String)₇₃, addTextNode(String)₇₄, getAllAttributes()₇₄, getAttributeValue(Name)₇₅, getChildElements(Name)₇₆, getChildElements(Name)₇₆, getElementName()₇₇, getEncodingStyle()₇₇, getNamespacePrefixes()₇₇, getNamespaceURI(String)₇₈, getVisibleNamespacePrefixes()₇₈, removeAttribute(Name)₇₈, removeContents()₇₉, removeNamespaceDeclaration(String)₇₉, setEncodingStyle(String)₇₉

Methods

addDetail()

```
public javax.xml.soap.Detail17 addDetail()  
    throws SOAPException
```

Creates an optional Detail object and sets it as the Detail object for this SOAPFault object.

It is illegal to add a detail when the fault already contains a detail. Therefore, this method should be called only after the existing detail has been removed.

Returns: the new Detail object

Throws:

[SOAPException₈₉](#) - if this SOAPFault object already contains a valid Detail object

addFaultReasonText(String, Locale)

```
public void addFaultReasonText(java.lang.String text, java.util.Locale locale)  
    throws SOAPException
```

Appends or replaces a Reason Text item containing the specified text message and an *xml:lang* derived from locale. If a Reason Text item with this *xml:lang* already exists its text value will be replaced with text. The locale parameter should not be null

Code sample:

```
SOAPFault fault = ...;  
fault.addFaultReasonText("Version Mismatch", Locale.ENGLISH);
```

Parameters:

text - reason message string

locale - Locale object representing the locale of the message

Throws:

[SOAPException₈₉](#) - if there was an error in adding the Reason text or the locale passed was null.

`java.lang.UnsupportedOperationException` - if this message does not support the SOAP 1.2 concept of Fault Reason.

Since: SAJ 1.3

appendFaultSubcode(QName)

```
public void appendFaultSubcode(javax.xml.namespace.QName subcode)  
    throws SOAPException
```

Adds a Subcode to the end of the sequence of Subcodes contained by this `SOAPFault`. Subcodes, which were introduced in SOAP 1.2, are represented by a recursive sequence of subelements rooted in the mandatory Code subelement of a SOAP Fault.

Parameters:

subcode - a QName containing the Value of the Subcode.

Throws:

[SOAPException](#)₈₉ - if there was an error in setting the Subcode

`java.lang.UnsupportedOperationException` - if this message does not support the SOAP 1.2 concept of Subcode.

Since: SAAJ 1.3

getDetail()

```
public javax.xml.soap.Detail17 getDetail()
```

Returns the optional detail element for this `SOAPFault` object.

A `Detail` object carries application-specific error information, the scope of the error information is restricted to faults in the `SOAPBodyElement` objects if this is a SOAP 1.1 Fault.

Returns: a `Detail` object with application-specific error information if present, null otherwise

getFaultActor()

```
public java.lang.String getFaultActor()
```

Gets the fault actor for this `SOAPFault` object.

If this `SOAPFault` supports SOAP 1.2 then this call is equivalent to [getFaultRole\(\)](#)₁₀₇

Returns: a `String` giving the actor in the message path that caused this `SOAPFault` object

See Also: [setFaultActor\(String\)](#)₁₀₈

getFaultCode()

```
public java.lang.String getFaultCode()
```

Gets the fault code for this `SOAPFault` object.

Returns: a `String` with the fault code

See Also: [getFaultCodeAsName\(\)](#)₁₀₄, [setFaultCode\(Name\)](#)₁₀₉

getFaultCodeAsName()

```
public javax.xml.soap.Name32 getFaultCodeAsName()
```

Gets the mandatory SOAP 1.1 fault code for this `SOAPFault` object as a `SAAJ Name` object. The SOAP 1.1 specification requires the value of the “faultcode” element to be of type `QName`. This method returns the content of the element as a `QName` in the form of a `SAAJ Name` object. This method should be used instead of the `getFaultCode` method since it allows applications to easily access the namespace name without additional parsing.

Returns: a `Name` representing the faultcode

Since: SAAJ 1.2

See Also: [setFaultCode\(Name\)](#)₁₀₉

getFaultCodeAsQName()

```
public javax.xml.namespace.QName getFaultCodeAsQName()
```

Gets the fault code for this `SOAPFault` object as a `QName` object.

Returns: a `QName` representing the faultcode

Since: SAAJ 1.3

See Also: [setFaultCode\(QName\)](#)₁₀₉

getFaultNode()

```
public java.lang.String getFaultNode()
```

Returns the optional `Node` element value for this `SOAPFault` object. The `Node` element is optional in SOAP 1.2.

Returns: Content of the `env:Fault/env:Node` element as a `String` or `null` if none

Throws:

`java.lang.UnsupportedOperationException` - if this message does not support the SOAP 1.2 concept of `Fault Node`.

Since: SAAJ 1.3

getFaultReasonLocales()

```
public java.util.Iterator<java.util.Locale> getFaultReasonLocales()  
    throws SOAPException
```

Returns an `Iterator` over a distinct sequence of `Locales` for which there are associated `Reason Text` items. Any of these `Locales` can be used in a call to `getFaultReasonText` in order to obtain a localized version of the `Reason Text` string.

Returns: an `Iterator` over a sequence of `Locale` objects for which there are associated `Reason Text` items.

Throws:

[SOAPException₈₉](#) - if there was an error in retrieving the fault Reason locales.

`java.lang.UnsupportedOperationException` - if this message does not support the SOAP 1.2 concept of Fault Reason.

Since: SAAJ 1.3

getFaultReasonText(Locale)

```
public java.lang.String getFaultReasonText(java.util.Locale locale)
    throws SOAPException
```

Returns the Reason Text associated with the given Locale. If more than one such Reason Text exists the first matching Text is returned

Parameters:

`locale` - the Locale for which a localized Reason Text is desired

Returns: the Reason Text associated with `locale`

Throws:

[SOAPException₈₉](#) - if there was an error in retrieving the fault Reason text for the specified locale.

`java.lang.UnsupportedOperationException` - if this message does not support the SOAP 1.2 concept of Fault Reason.

Since: SAAJ 1.3

See Also: [getFaultString\(\)₁₀₇](#)

getFaultReasonTexts()

```
public java.util.Iterator<String> getFaultReasonTexts()
    throws SOAPException
```

Returns an Iterator over a sequence of String objects containing all of the Reason Text items for this SOAPFault.

Returns: an Iterator over env:Fault/env:Reason/env:Text items.

Throws:

[SOAPException₈₉](#) - if there was an error in retrieving the fault Reason texts.

`java.lang.UnsupportedOperationException` - if this message does not support the SOAP 1.2 concept of Fault Reason.

Since: SAAJ 1.3

getFaultRole()

```
public java.lang.String getFaultRole()
```

Returns the optional Role element value for this SOAPFault object. The Role element is optional in SOAP 1.2.

Returns: Content of the env:Fault/env:Role element as a String or null if none

Throws:

java.lang.UnsupportedOperationException - if this message does not support the SOAP 1.2 concept of Fault Role.

Since: SAAJ 1.3

getFaultString()

```
public java.lang.String getFaultString()
```

Gets the fault string for this SOAPFault object.

If this SOAPFault is part of a message that supports SOAP 1.2 then this call is equivalent to:

```
String reason = null;
try {
    reason = (String) getFaultReasonTexts().next();
} catch (SOAPException e) {}
return reason;
```

Returns: a String giving an explanation of the fault

See Also: [setFaultString\(String\)](#)₁₁₁, [setFaultString\(String, Locale\)](#)₁₁₁

getFaultStringLocale()

```
public java.util.Locale getFaultStringLocale()
```

Gets the locale of the fault string for this SOAPFault object.

If this SOAPFault is part of a message that supports SOAP 1.2 then this call is equivalent to:

```
Locale locale = null;
try {
    locale = (Locale) getFaultReasonLocales().next();
} catch (SOAPException e) {}
return locale;
```

Returns: a Locale object indicating the native language of the fault string or null if no locale was specified

Since: SAAJ 1.2

See Also: [setFaultString\(String, Locale\)](#)₁₁₁

getFaultSubcodes()

```
public java.util.Iterator<javax.xml.namespace.QName> getFaultSubcodes()
```

Gets the Subcodes for this SOAPFault as an iterator over QNames.

Returns: an Iterator that accesses a sequence of QNames. This Iterator should not support the optional remove method. The order in which the Subcodes are returned reflects the hierarchy of Subcodes present in the fault from top to bottom.

Throws:

`java.lang.UnsupportedOperationException` - if this message does not support the SOAP 1.2 concept of Subcode.

Since: SAAJ 1.3

hasDetail()

```
public boolean hasDetail()
```

Returns true if this SOAPFault has a Detail subelement and false otherwise. Equivalent to `(getDetail() != null)`.

Returns: true if this SOAPFault has a Detail subelement and false otherwise.

Since: SAAJ 1.3

removeAllFaultSubcodes()

```
public void removeAllFaultSubcodes()
```

Removes any Subcodes that may be contained by this SOAPFault. Subsequent calls to `getFaultSubcodes` will return an empty iterator until a call to `appendFaultSubcode` is made.

Throws:

`java.lang.UnsupportedOperationException` - if this message does not support the SOAP 1.2 concept of Subcode.

Since: SAAJ 1.3

setFaultActor(String)

```
public void setFaultActor(java.lang.String faultActor)  
    throws SOAPException
```

Sets this SOAPFault object with the given fault actor.

The fault actor is the recipient in the message path who caused the fault to happen.

If this SOAPFault supports SOAP 1.2 then this call is equivalent to

[setFaultRole\(String\)](#) 111-

Parameters:

`faultActor` - a `String` identifying the actor that caused this `SOAPFault` object

Throws:

`SOAPException`₈₉ - if there was an error in adding the `faultActor` to the underlying XML tree.

See Also: `getFaultActor()`₁₀₄

setFaultCode(Name)

```
public void setFaultCode(javax.xml.soap.Name32 faultCodeQName)
    throws SOAPException
```

Sets this `SOAPFault` object with the given fault code.

Fault codes, which give information about the fault, are defined in the SOAP 1.1 specification. A fault code is mandatory and must be of type `QName`. This method provides a convenient way to set a fault code. For example,

```
SOAPEnvelope se = ...;
// Create a qualified name in the SOAP namespace with a localName
// of "Client". Note that prefix parameter is optional and is null
// here which causes the implementation to use an appropriate prefix.
Name qname = se.createName("Client", null,
                           SOAPConstants.URI_NS_SOAP_ENVELOPE);

SOAPFault fault = ...;
fault.setFaultCode(qname);
```

It is preferable to use this method over `setFaultCode(String)`₁₁₀.

Parameters:

`faultCodeQName` - a `Name` object giving the fault code to be set. It must be namespace qualified.

Throws:

`SOAPException`₈₉ - if there was an error in adding the `faultCode` element to the underlying XML tree.

Since: SAAJ 1.2

See Also: `getFaultCodeAsName()`₁₀₄

setFaultCode(QName)

```
public void setFaultCode(javax.xml.namespace.QName faultCodeQName)
    throws SOAPException
```

Sets this `SOAPFault` object with the given fault code. It is preferable to use this method over `setFaultCode(Name)`₁₀₉.

Parameters:

`faultCodeQName` - a `QName` object giving the fault code to be set. It must be namespace qualified.

Throws:

[SOAPException₈₉](#) - if there was an error in adding the *faultcode* element to the underlying XML tree.

Since: SAAJ 1.3

See Also: [getFaultCodeAsQName\(\)](#)₁₀₅, [setFaultCode\(Name\)](#)₁₀₉,
[getFaultCodeAsQName\(\)](#)₁₀₅

setFaultCode(String)

```
public void setFaultCode(java.lang.String faultCode)
    throws SOAPException
```

Sets this `SOAPFault` object with the give fault code.

Fault codes, which given information about the fault, are defined in the SOAP 1.1 specification. This element is mandatory in SOAP 1.1. Because the fault code is required to be a `QName` it is preferable to use the [setFaultCode\(Name\)](#)₁₀₉ form of this method.

Parameters:

`faultCode` - a `String` giving the fault code to be set. It must be of the form “prefix:localName” where the prefix has been defined in a namespace declaration.

Throws:

[SOAPException₈₉](#) - if there was an error in adding the `faultCode` to the underlying XML tree.

See Also: [setFaultCode\(Name\)](#)₁₀₉, [getFaultCode\(\)](#)₁₀₄,
[SOAPElement.addNamespaceDeclaration\(String, String\)](#)₇₃

setFaultNode(String)

```
public void setFaultNode(java.lang.String uri)
    throws SOAPException
```

Creates or replaces any existing `Node` element value for this `SOAPFault` object. The `Node` element is optional in SOAP 1.2.

Parameters:

`uri` - the URI of the `Node`

Throws:

[SOAPException₈₉](#) - if there was an error in setting the `Node` for this `SOAPFault` object.

`java.lang.UnsupportedOperationException` - if this message does not support the SOAP 1.2 concept of Fault Node.

Since: SAAJ 1.3

setFaultRole(String)

```
public void setFaultRole(java.lang.String uri)
           throws SOAPException
```

Creates or replaces any existing Role element value for this `SOAPFault` object. The Role element is optional in SOAP 1.2.

Parameters:

`uri` - the URI of the Role

Throws:

`SOAPException99` - if there was an error in setting the Role for this `SOAPFault` object.

`java.lang.UnsupportedOperationException` - if this message does not support the SOAP 1.2 concept of Fault Role.

Since: SAAJ 1.3

setFaultString(String)

```
public void setFaultString(java.lang.String faultString)
           throws SOAPException
```

Sets the fault string for this `SOAPFault` object to the given string.

If this `SOAPFault` is part of a message that supports SOAP 1.2 then this call is equivalent to:

```
addFaultReasonText(faultString, Locale.getDefault());
```

Parameters:

`faultString` - a `String` giving an explanation of the fault

Throws:

`SOAPException99` - if there was an error in adding the `faultString` to the underlying XML tree.

See Also: `getFaultString()107`

setFaultString(String, Locale)

```
public void setFaultString(java.lang.String faultString, java.util.Locale locale)
           throws SOAPException
```

Sets the fault string for this `SOAPFault` object to the given string and localized to the given locale.

If this `SOAPFault` is part of a message that supports SOAP 1.2 then this call is equivalent to:

```
addFaultReasonText(faultString, locale);
```

Parameters:

`faultString` - a `String` giving an explanation of the fault

`locale` - a `Locale` object indicating the native language of the `faultString`

Throws:

[SOAPException](#)₈₉ - if there was an error in adding the `faultString` to the underlying XML tree.

Since: SAAJ 1.2

See Also: [getFaultString\(\)](#)₁₀₇

2.23 SOAPFaultElement

Declaration

```
public interface SOAPFaultElement extends SOAPElement67
```

All Superinterfaces: org.w3c.dom.Element, org.w3c.dom.Node, Node₃₅, SOAPElement₆₇

All Known Subinterfaces: Detail₁₇

Description

A representation of the contents in a SOAPFault object. The Detail interface is a SOAPFaultElement.

Content is added to a SOAPFaultElement using the SOAPElement method addTextNode.

Inherited Member Summary

Fields inherited from interface Node

ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE

Methods inherited from interface Element

getAttribute(String), getAttributeNS(String, String), getAttributeNode(String), getAttributeNodeNS(String, String), getElementsByTagName(String), getElementsByTagNameNS(String, String), getTagName(), hasAttribute(String), hasAttributeNS(String, String), removeAttribute(String), removeAttributeNS(String, String), removeAttributeNode(Attr), setAttribute(String, String), setAttributeNS(String, String, String), setAttributeNode(Attr), setAttributeNodeNS(Attr)

Methods inherited from interface Node₃₅

Inherited Member Summary

`detachNode()`₃₆, `getParentElement()`₃₆, `getValue()`₃₆, `recycleNode()`₃₇,
`setParentElement(SOAPElement)`₃₇, `setValue(String)`₃₇

Methods inherited from interface **Node**

`appendChild(Node)`, `cloneNode(boolean)`, `getAttributes()`, `getChildNodes()`,
`getFirstChild()`, `getLastChild()`, `getLocalName()`, `getNamespaceURI()`, `getNextSibling()`,
`getNodeName()`, `getNodeType()`, `getNodeValue()`, `getOwnerDocument()`, `getParentNode()`,
`getPrefix()`, `getPreviousSibling()`, `hasAttributes()`, `hasChildNodes()`,
`insertBefore(Node, Node)`, `isSupported(String, String)`, `normalize()`,
`removeChild(Node)`, `replaceChild(Node, Node)`, `setNodeValue(String)`, `setPrefix(String)`

Methods inherited from interface **SOAPElement**₆₇

`addAttribute(Name, String)`₇₀, `addChildElement(SOAPElement)`₇₂,
`addChildElement(SOAPElement)`₇₂, `addChildElement(SOAPElement)`₇₂,
`addChildElement(SOAPElement)`₇₂, `addChildElement(SOAPElement)`₇₂,
`addNamespaceDeclaration(String, String)`₇₃, `addTextNode(String)`₇₄,
`getAllAttributes()`₇₄, `getAttributeValue(Name)`₇₅, `getChildElements(Name)`₇₆,
`getChildElements(Name)`₇₆, `getElementName()`₇₇, `getEncodingStyle()`₇₇,
`getNamespacePrefixes()`₇₇, `getNamespaceURI(String)`₇₈, `getVisibleNamespacePrefixes()`₇₈,
`removeAttribute(Name)`₇₈, `removeContents()`₇₉, `removeNamespaceDeclaration(String)`₇₉,
`setEncodingStyle(String)`₇₉

2.24 SOAPHeader

Declaration

```
public interface SOAPHeader extends SOAPElement67
```

All Superinterfaces: `org.w3c.dom.Element`, `org.w3c.dom.Node`, `Node35`, `SOAPElement67`

Description

A representation of the SOAP header element. A SOAP header element consists of XML data that affects the way the application-specific content is processed by the message provider. For example, transaction semantics, authentication information, and so on, can be specified as the content of a SOAPHeader object.

A SOAPEnvelope object contains an empty SOAPHeader object by default. If the SOAPHeader object, which is optional, is not needed, it can be retrieved and deleted with the following line of code. The variable *se* is a SOAPEnvelope object.

```
se.getHeader().detachNode();
```

A SOAPHeader object is created with the SOAPEnvelope method `addHeader`. This method, which creates a new header and adds it to the envelope, may be called only after the existing header has been removed.

```
se.getHeader().detachNode();  
SOAPHeader sh = se.addHeader();
```

A SOAPHeader object can have only SOAPHeaderElement objects as its immediate children. The method `addHeaderElement` creates a new HeaderElement object and adds it to the SOAPHeader object. In the following line of code, the argument to the method `addHeaderElement` is a Name object that is the name for the new HeaderElement object.

```
SOAPHeaderElement shElement = sh.addHeaderElement(name);
```

See Also: `SOAPHeaderElement123`

Member Summary	
Methods	
SOAPHeaderElement	addElement(Name name) ¹¹⁸ Creates a new SOAPHeaderElement object initialized with the specified name and adds it to this SOAPHeader object.
SOAPHeaderElement	addElement(javax.xml.namespace.QName qname) ¹¹⁸ Creates a new SOAPHeaderElement object initialized with the specified qname and adds it to this SOAPHeader object.
SOAPHeaderElement	addNotUnderstoodHeaderElement(javax.xml.namespace.QName name) ¹¹⁸ Creates a new NotUnderstood SOAPHeaderElement object initialized with the specified name and adds it to this SOAPHeader object.
SOAPHeaderElement	addUpgradeHeaderElement(java.util.Iterator<java.lang.String> supportedSOAPURIs) ¹¹⁹ Creates a new Upgrade SOAPHeaderElement object initialized with the specified List of supported SOAP URIs and adds it to this SOAPHeader object.
SOAPHeaderElement	addUpgradeHeaderElement(java.lang.String[] supportedSoapUri) ¹¹⁹ Creates a new Upgrade SOAPHeaderElement object initialized with the specified supported SOAP URI and adds it to this SOAPHeader object.
SOAPHeaderElement	addUpgradeHeaderElement(java.lang.String supportedSoapUris) ¹²⁰ Creates a new Upgrade SOAPHeaderElement object initialized with the specified array of supported SOAP URIs and adds it to this SOAPHeader object.
java.util.Iterator<SOAPHeaderElement>	examineAllHeaderElements() ¹²⁰ Returns an Iterator over all the SOAPHeaderElement objects in this SOAPHeader object.
java.util.Iterator<SOAPHeaderElement>	examineHeaderElements(java.lang.String actor) ¹²⁰ Returns an Iterator over all the SOAPHeaderElement objects in this SOAPHeader object that have the specified actor.
java.util.Iterator<SOAPHeaderElement>	examineMustUnderstandHeaderElements(java.lang.String actor) ¹²¹ Returns an Iterator over all the SOAPHeaderElement objects in this SOAPHeader object that have the specified actor and that have a MustUnderstand attribute whose value is equivalent to true.
java.util.Iterator<SOAPHeaderElement>	extractAllHeaderElements() ¹²¹ Returns an Iterator over all the SOAPHeaderElement objects in this SOAPHeader object and detaches them from this SOAPHeader object.
java.util.Iterator<SOAPHeaderElement>	extractHeaderElements(java.lang.String actor) ¹²¹ Returns an Iterator over all the SOAPHeaderElement objects in this SOAPHeader object that have the specified actor and detaches them from this SOAPHeader object.

Inherited Member Summary
Fields inherited from interface Node

Inherited Member Summary

ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE

Methods inherited from interface **Element**

getAttribute(String), getAttributeNS(String, String), getAttributeNode(String), getAttributeNodeNS(String, String), getElementsByTagName(String), getElementsByTagNameNS(String, String), getTagName(), hasAttribute(String), hasAttributeNS(String, String), removeAttribute(String), removeAttributeNS(String, String), removeAttributeNode(Attr), setAttribute(String, String), setAttributeNS(String, String, String), setAttributeNode(Attr), setAttributeNodeNS(Attr)

Methods inherited from interface **Node₃₅**

detachNode()₃₆, getParentElement()₃₆, getValue()₃₆, recycleNode()₃₇, setParentElement(SOAPElement)₃₇, setValue(String)₃₇

Methods inherited from interface **Node**

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getLocalName(), getNamespaceURI(), getNextSibling(), getNodeName(), getNodeType(), getNodeValue(), getOwnerDocument(), getParentNode(), getPrefix(), getPreviousSibling(), hasAttributes(), hasChildNodes(), insertBefore(Node, Node), isSupported(String, String), normalize(), removeChild(Node), replaceChild(Node, Node), setNodeValue(String), setPrefix(String)

Methods inherited from interface **SOAPElement₆₇**

addAttribute(Name, String)₇₀, addChildElement(SOAPElement)₇₂, addChildElement(SOAPElement)₇₂, addChildElement(SOAPElement)₇₂, addChildElement(SOAPElement)₇₂, addNamespaceDeclaration(String, String)₇₃, addTextNode(String)₇₄, getAllAttributes()₇₄, getAttributeValue(Name)₇₅, getChildElements(Name)₇₆, getChildElements(Name)₇₆, getElementName()₇₇, getEncodingStyle()₇₇, getNamespacePrefixes()₇₇, getNamespaceURI(String)₇₈, getVisibleNamespacePrefixes()₇₈, removeAttribute(Name)₇₈, removeContents()₇₉, removeNamespaceDeclaration(String)₇₉, setEncodingStyle(String)₇₉

Methods

addHeaderElement(Name)

```
public javax.xml.soap.SOAPHeaderElement123 addHeaderElement(javax.xml.soap.Name32
    name)
    throws SOAPException
```

Creates a new SOAPHeaderElement object initialized with the specified name and adds it to this SOAPHeader object.

Parameters:

name - a Name object with the name of the new SOAPHeaderElement object

Returns: the new SOAPHeaderElement object that was inserted into this SOAPHeader object

Throws:

SOAPException₈₉ - if a SOAP error occurs

addHeaderElement(QName)

```
public javax.xml.soap.SOAPHeaderElement123
    addHeaderElement(javax.xml.namespace.QName qname)
    throws SOAPException
```

Creates a new SOAPHeaderElement object initialized with the specified qname and adds it to this SOAPHeader object.

Parameters:

qname - a QName object with the qname of the new SOAPHeaderElement object

Returns: the new SOAPHeaderElement object that was inserted into this SOAPHeader object

Throws:

SOAPException₈₉ - if a SOAP error occurs

Since: SAAJ 1.3

See Also: [addHeaderElement \(Name\)](#) ₁₁₈

addNotUnderstoodHeaderElement(QName)

```
public javax.xml.soap.SOAPHeaderElement123
    addNotUnderstoodHeaderElement(javax.xml.namespace.QName name)
    throws SOAPException
```

Creates a new NotUnderstood SOAPHeaderElement object initialized with the specified name and adds it to this SOAPHeader object. This operation is supported only by SOAP 1.2.

Parameters:

name - a QName object with the name of the SOAPHeaderElement object that was not understood.

Returns: the new SOAPHeaderElement object that was inserted into this SOAPHeader object

Throws:

SOAPException₈₉ - if a SOAP error occurs.

java.lang.UnsupportedOperationException - if this is a SOAP 1.1 Header.

Since: SAAJ 1.3

addUpgradeHeaderElement(Iterator<String>)

```
public javax.xml.soap.SOAPHeaderElement123
    addUpgradeHeaderElement(java.util.Iterator<String> supportedSOAPURIs)
        throws SOAPException
```

Creates a new Upgrade SOAPHeaderElement object initialized with the specified List of supported SOAP URIs and adds it to this SOAPHeader object. This operation is supported on both SOAP 1.1 and SOAP 1.2 header.

Parameters:

supportedSOAPURIs - an Iterator object with the URIs of SOAP versions supported.

Returns: the new SOAPHeaderElement object that was inserted into this SOAPHeader object

Throws:

SOAPException₈₉ - if a SOAP error occurs.

Since: SAAJ 1.3

addUpgradeHeaderElement(String)

```
public javax.xml.soap.SOAPHeaderElement123 addUpgradeHeaderElement(java.lang.String
    supportedSoapUri)
    throws SOAPException
```

Creates a new Upgrade SOAPHeaderElement object initialized with the specified supported SOAP URI and adds it to this SOAPHeader object. This operation is supported on both SOAP 1.1 and SOAP 1.2 header.

Parameters:

supportedSoapUri - the URI of SOAP the version that is supported.

Returns: the new SOAPHeaderElement object that was inserted into this SOAPHeader object

Throws:

SOAPException₈₉ - if a SOAP error occurs.

Since: SAAJ 1.3

addUpgradeHeaderElement(String[])

```
public javax.xml.soap.SOAPHeaderElement123
    addUpgradeHeaderElement(java.lang.String[] supportedSoapUris)
    throws SOAPException
```

Creates a new Upgrade SOAPHeaderElement object initialized with the specified array of supported SOAP URIs and adds it to this SOAPHeader object. This operation is supported on both SOAP 1.1 and SOAP 1.2 header.

Parameters:

supportedSoapUris - an array of the URIs of SOAP versions supported.

Returns: the new SOAPHeaderElement object that was inserted into this SOAPHeader object

Throws:

SOAPException₈₉ - if a SOAP error occurs.

Since: SAAJ 1.3

examineAllHeaderElements()

```
public java.util.Iterator<SOAPHeaderElement> examineAllHeaderElements()
```

Returns an Iterator over all the SOAPHeaderElement objects in this SOAPHeader object.

Returns: an Iterator object over all the SOAPHeaderElement objects contained by this SOAPHeader

Since: SAAJ 1.2

See Also: [extractAllHeaderElements\(\)](#)₁₂₁

examineHeaderElements(String)

```
public java.util.Iterator<SOAPHeaderElement>
    examineHeaderElements(java.lang.String actor)
```

Returns an Iterator over all the SOAPHeaderElement objects in this SOAPHeader object that have the specified actor. An actor is a global attribute that indicates the intermediate parties that should process a message before it reaches its ultimate receiver. An actor receives the message and processes it before sending it on to the next actor. The default actor is the ultimate intended recipient for the message, so if no actor attribute is included in a SOAPHeader object, it is sent to the ultimate receiver along with the message body.

In SOAP 1.2 the *env:actor* attribute is replaced by the *env:role* attribute, but with essentially the same semantics.

Parameters:

actor - a String giving the URI of the actor/role for which to search

Returns: an Iterator object over all the SOAPHeaderElement objects that contain the specified actor/role

See Also: [extractHeaderElements\(String\)](#)¹²¹,
[SOAPConstants.URI_SOAP_ACTOR_NEXT](#)⁶⁶

examineMustUnderstandHeaderElements(String)

```
public java.util.Iterator<SOAPHeaderElement>  
    examineMustUnderstandHeaderElements(java.lang.String actor)
```

Returns an Iterator over all the SOAPHeaderElement objects in this SOAPHeader object that have the specified actor and that have a MustUnderstand attribute whose value is equivalent to true.

In SOAP 1.2 the *env:actor* attribute is replaced by the *env:role* attribute, but with essentially the same semantics.

Parameters:

actor - a String giving the URI of the actor/role for which to search

Returns: an Iterator object over all the SOAPHeaderElement objects that contain the specified actor/role and are marked as MustUnderstand

Since: SAAJ 1.2

See Also: [examineHeaderElements\(String\)](#)¹²⁰,
[extractHeaderElements\(String\)](#)¹²¹,
[SOAPConstants.URI_SOAP_ACTOR_NEXT](#)⁶⁶

extractAllHeaderElements()

```
public java.util.Iterator<SOAPHeaderElement> extractAllHeaderElements()
```

Returns an Iterator over all the SOAPHeaderElement objects in this SOAPHeader object and detaches them from this SOAPHeader object.

Returns: an Iterator object over all the SOAPHeaderElement objects contained by this SOAPHeader

Since: SAAJ 1.2

See Also: [examineAllHeaderElements\(\)](#)¹²⁰

extractHeaderElements(String)

```
public java.util.Iterator<SOAPHeaderElement>  
    extractHeaderElements(java.lang.String actor)
```


Returns an `Iterator` over all the `SOAPHeaderElement` objects in this `SOAPHeader` object that have the specified actor and detaches them from this `SOAPHeader` object.

This method allows an actor to process the parts of the `SOAPHeader` object that apply to it and to remove them before passing the message on to the next actor.

In SOAP 1.2 the *env:actor* attribute is replaced by the *env:role* attribute, but with essentially the same semantics.

Parameters:

actor - a `String` giving the URI of the actor/role for which to search

Returns: an `Iterator` object over all the `SOAPHeaderElement` objects that contain the specified actor/role

See Also: [examineHeaderElements\(String\)](#)₁₂₀,
[SOAPConstants.URI_SOAP_ACTOR_NEXT](#)₆₆

2.25 SOAPHeaderElement

Declaration

```
public interface SOAPHeaderElement extends SOAPElement67
```

All Superinterfaces: org.w3c.dom.Element, org.w3c.dom.Node, Node₃₅, SOAPElement₆₇

Description

An object representing the contents in the SOAP header part of the SOAP envelope. The immediate children of a SOAPHeader object can be represented only as SOAPHeaderElement objects.

A SOAPHeaderElement object can have other SOAPElement objects as its children.

Member Summary

Methods

java.lang.String	getActor() ₁₂₅	Returns the uri of the actor associated with this SOAPHeaderElement object.
boolean	getMustUnderstand() ₁₂₅	Returns whether the mustUnderstand attribute for this SOAPHeaderElement object is turned on.
boolean	getRelay() ₁₂₅	Returns the boolean value of the <i>relay</i> attribute for this SOAPHeaderElement
java.lang.String	getRole() ₁₂₆	Returns the value of the <i>Role</i> attribute of this SOAPHeaderElement.
void	setActor(java.lang.String actorURI) ₁₂₆	Sets the actor associated with this SOAPHeaderElement object to the specified actor.
void	setMustUnderstand(boolean mustUnderstand) ₁₂₆	Sets the mustUnderstand attribute for this SOAPHeaderElement object to be on or off.
void	setRelay(boolean relay) ₁₂₇	Sets the <i>relay</i> attribute for this SOAPHeaderElement to be either true or false.

Member Summary

void [setRole\(java.lang.String uri\)](#) ¹²⁷

Sets the Role associated with this SOAPHeaderElement object to the specified Role.

Inherited Member Summary

Fields inherited from interface Node

ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE

Methods inherited from interface Element

getAttribute(String), getAttributeNS(String, String), getAttributeNode(String), getAttributeNodeNS(String, String), getElementsByTagName(String), getElementsByTagNameNS(String, String), getTagName(), hasAttribute(String), hasAttributeNS(String, String), removeAttribute(String), removeAttributeNS(String, String), removeAttributeNode(Attr), setAttribute(String, String), setAttributeNS(String, String, String), setAttributeNode(Attr), setAttributeNodeNS(Attr)

Methods inherited from interface [Node](#) ³⁵

[detachNode\(\)](#) ³⁶, [getParentElement\(\)](#) ³⁶, [getValue\(\)](#) ³⁶, [recycleNode\(\)](#) ³⁷, [setParentElement\(SOAPElement\)](#) ³⁷, [setValue\(String\)](#) ³⁷

Methods inherited from interface Node

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(), getFirstChild(), getLastChild(), getLocalName(), getNamespaceURI(), getNextSibling(), getNodeName(), getNodeType(), getNodeValue(), getOwnerDocument(), getParentNode(), getPrefix(), getPreviousSibling(), hasAttributes(), hasChildNodes(), insertBefore(Node, Node), isSupported(String, String), normalize(), removeChild(Node), replaceChild(Node, Node), setNodeValue(String), setPrefix(String)

Methods inherited from interface [SOAPElement](#) ⁶⁷

Inherited Member Summary

```
addAttribute(Name, String) 70, addChildElement(SOAPElement) 72,  
addChildElement(SOAPElement) 72, addChildElement(SOAPElement) 72,  
addChildElement(SOAPElement) 72, addChildElement(SOAPElement) 72,  
addNamespaceDeclaration(String, String) 73, addTextNode(String) 74,  
getAllAttributes() 74, getAttributeValue(Name) 75, getChildElements(Name) 76,  
getChildElements(Name) 76, getElementName() 77, getEncodingStyle() 77,  
getNamespacePrefixes() 77, getNamespaceURI(String) 78, getVisibleNamespacePrefixes() 78,  
removeAttribute(Name) 78, removeContents() 79, removeNamespaceDeclaration(String) 79,  
setEncodingStyle(String) 79
```

Methods

getActor()

```
public java.lang.String getActor()
```

Returns the uri of the actor associated with this SOAPHeaderElement object.

If this SOAPHeaderElement supports SOAP 1.2 then this call is equivalent to [getRole\(\)](#) ¹²⁶-

Returns: a String giving the URI of the actor

See Also: [setActor\(String\)](#) ¹²⁶

getMustUnderstand()

```
public boolean getMustUnderstand()
```

Returns whether the mustUnderstand attribute for this SOAPHeaderElement object is turned on.

Returns: true if the mustUnderstand attribute of this SOAPHeaderElement object is turned on; false otherwise

getRelay()

```
public boolean getRelay()
```

Returns the boolean value of the *relay* attribute for this SOAPHeaderElement

Returns: true if the relay attribute is turned on; false otherwise

Throws:

`java.lang.UnsupportedOperationException` - if this message does not support the SOAP 1.2 concept of Relay attribute.

Since: SAAJ 1.3

See Also: [getMustUnderstand\(\)](#) ¹²⁵, [setRelay\(boolean\)](#) ¹²⁷

getRole()

```
public java.lang.String getRole()
```

Returns the value of the *Role* attribute of this SOAPHeaderElement.

Returns: a String giving the URI of the Role

Throws:

java.lang.UnsupportedOperationException - if this message does not support the SOAP 1.2 concept of Fault Role.

Since: SAAJ 1.3

setActor(String)

```
public void setActor(java.lang.String actorURI)
```

Sets the actor associated with this SOAPHeaderElement object to the specified actor. The default value of an actor is: SOAPConstants.URI_SOAP_ACTOR_NEXT

If this SOAPHeaderElement supports SOAP 1.2 then this call is equivalent to

[setRole\(String\)](#) ₁₂₇

Parameters:

actorURI - a String giving the URI of the actor to set

Throws:

java.lang.IllegalArgumentException - if there is a problem in setting the actor.

See Also: [getActor\(\)](#) ₁₂₅

setMustUnderstand(boolean)

```
public void setMustUnderstand(boolean mustUnderstand)
```

Sets the mustUnderstand attribute for this SOAPHeaderElement object to be on or off.

If the mustUnderstand attribute is on, the actor who receives the SOAPHeaderElement must process it correctly. This ensures, for example, that if the SOAPHeaderElement object modifies the message, that the message is being modified correctly.

Parameters:

mustUnderstand - true to set the mustUnderstand attribute on; false to turn it off

Throws:

java.lang.IllegalArgumentException - if there is a problem in setting the mustUnderstand attribute

See Also: [getMustUnderstand\(\)](#) ₁₂₅

setRelay(boolean)

```
public void setRelay(boolean relay)
```

```
throws SOAPException
```

Sets the *relay* attribute for this SOAPHeaderElement to be either true or false.

The SOAP relay attribute is set to true to indicate that the SOAP header block must be relayed by any node that is targeted by the header block but not actually process it. This attribute is ignored on header blocks whose mustUnderstand attribute is set to true or that are targeted at the ultimate receiver (which is the default). The default value of this attribute is false.

Parameters:

relay - the new value of the *relay* attribute

Throws:

[SOAPException₈₉](#) - if there is a problem in setting the relay attribute.

[java.lang.UnsupportedOperationException](#) - if this message does not support the SOAP 1.2 concept of Relay attribute.

Since: SAAJ 1.3

See Also: [setMustUnderstand\(boolean\)₁₂₆](#), [getRelay\(\)₁₂₅](#)

setRole(String)

```
public void setRole(java.lang.String uri)
```

```
throws SOAPException
```

Sets the Role associated with this SOAPHeaderElement object to the specified Role.

Parameters:

uri - the URI of the Role

Throws:

[SOAPException₈₉](#) - if there is an error in setting the role

[java.lang.UnsupportedOperationException](#) - if this message does not support the SOAP 1.2 concept of Fault Role.

Since: SAAJ 1.3

2.26 SOAPMessage

Declaration

```
public abstract class SOAPMessage
```

```
java.lang.Object
```

```
|  
+-- javax.xml.soap.SOAPMessage
```

Description

The root class for all SOAP messages. As transmitted on the “wire”, a SOAP message is an XML document or a MIME message whose first body part is an XML/SOAP document.

A SOAPMessage object consists of a SOAP part and optionally one or more attachment parts. The SOAP part for a SOAPMessage object is a SOAPPart object, which contains information used for message routing and identification, and which can contain application-specific content. All data in the SOAP Part of a message must be in XML format.

A new SOAPMessage object contains the following by default:

- A SOAPPart object
- A SOAPEnvelope object
- A SOAPBody object
- A SOAPHeader object

The SOAP part of a message can be retrieved by calling the method `SOAPMessage.getSOAPPart()`. The `SOAPEnvelope` object is retrieved from the `SOAPPart` object, and the `SOAPEnvelope` object is used to retrieve the `SOAPBody` and `SOAPHeader` objects.

```
SOAPPart sp = message.getSOAPPart();  
SOAPEnvelope se = sp.getEnvelope();  
SOAPBody sb = se.getBody();  
SOAPHeader sh = se.getHeader();
```

In addition to the mandatory `SOAPPart` object, a `SOAPMessage` object may contain zero or more `AttachmentPart` objects, each of which contains application-specific data. The `SOAPMessage` interface provides methods for creating `AttachmentPart` objects and also for adding them to a `SOAPMessage` object. A party that has received a `SOAPMessage` object can examine its contents by retrieving individual attachment parts.

Unlike the rest of a SOAP message, an attachment is not required to be in XML format and can therefore be anything from simple text to an image file. Consequently, any message content that is not in XML format must be in an `AttachmentPart` object.

A `MessageFactory` object may create `SOAPMessage` objects with behavior that is specialized to a particular implementation or application of SAAJ. For instance, a `MessageFactory` object may produce `SOAPMessage` objects that conform to a particular Profile such as ebXML. In this case a `MessageFactory` object might produce `SOAPMessage` objects that are initialized with ebXML headers.

In order to ensure backward source compatibility, methods that are added to this class after version 1.1 of the SAAJ specification are all concrete instead of abstract and they all have default implementations. Unless otherwise noted in the JavaDocs for those methods the default implementations simply throw an `UnsupportedOperationException` and the SAAJ implementation code must override them with methods that provide the specified behavior. Legacy client code does not have this restriction, however, so long as there is no claim made that it conforms to some later version of the specification than it was originally written for. A legacy class that extends the `SOAPMessage` class can be compiled and/or run against succeeding versions of the SAAJ API without modification. If such a class was correctly implemented then it will continue to behave correctly relative to the version of the specification against which it was written.

See Also: [MessageFactory₂₂](#), [AttachmentPart₅](#)

Member Summary		
Fields		
static	CHARACTER_SET_ENCODING₁₃₁	
java.lang.String		Specifies the character type encoding for the SOAP Message.
static	WRITE_XML_DECLARATION₁₃₁	
java.lang.String		Specifies whether the SOAP Message will contain an XML declaration when it is sent.
Constructors		
	SOAPMessage ()₁₃₁	
Methods		
abstract void	addAttachmentPart (AttachmentPart attachmentPart)₁₃₁	Adds the given AttachmentPart object to this SOAPMessage object.
abstract int	countAttachments ()₁₃₂	Gets a count of the number of attachments in this message.
abstract	createAttachmentPart ()₁₃₂	
AttachmentPart		Creates a new empty AttachmentPart object.
AttachmentPart	createAttachmentPart (DataHandler dataHandler)₁₃₂	Creates an AttachmentPart object and populates it using the given DataHandler object.

Member Summary

AttachmentPart	createAttachmentPart(java.lang.Object content, java.lang.String contentType) ¹³³	Creates an AttachmentPart object and populates it with the specified data of the specified content type.
abstract AttachmentPart	getAttachment(SOAPElement element) ¹³³	Returns an AttachmentPart object that is associated with an attachment that is referenced by this SOAPElement or null if no such attachment exists.
abstract java.util.Iterator<AttachmentPart>	getAttachments() ¹³⁴	Retrieves all the AttachmentPart objects that are part of this SOAPMessage object.
abstract java.util.Iterator<AttachmentPart>	getAttachments(MimeHeaders headers) ¹³⁴	Retrieves all the AttachmentPart objects that have header entries that match the specified headers.
abstract java.lang.String	getContentDescription() ¹³⁴	Retrieves a description of this SOAPMessage object's content.
abstract MimeHeaders	getMimeHeaders() ¹³⁴	Returns all the transport-specific MIME headers for this SOAPMessage object in a transport-independent fashion.
java.lang.Object	getProperty(java.lang.String property) ¹³⁵	Retrieves value of the specified property.
SOAPBody	getSOAPBody() ¹³⁵	Gets the SOAP Body contained in this SOAPMessage object.
SOAPHeader	getSOAPHeader() ¹³⁵	Gets the SOAP Header contained in this SOAPMessage object.
abstract SOAPPart	getSOAPPart() ¹³⁵	Gets the SOAP part of this SOAPMessage object.
abstract void	removeAllAttachments() ¹³⁶	Removes all AttachmentPart objects that have been added to this SOAPMessage object.
abstract void	removeAttachments(MimeHeaders headers) ¹³⁶	Removes all the AttachmentPart objects that have header entries that match the specified headers.
abstract void	saveChanges() ¹³⁶	Updates this SOAPMessage object with all the changes that have been made to it.
abstract boolean	saveRequired() ¹³⁶	Indicates whether this SOAPMessage object needs to have the method saveChanges called on it.
abstract void	setContentDescription(java.lang.String description) ¹³⁷	Sets the description of this SOAPMessage object's content with the given description.
void	setProperty(java.lang.String property, java.lang.Object value) ¹³⁷	Associates the specified value with the specified property.
abstract void	writeTo(java.io.OutputStream out) ¹³⁸	Writes this SOAPMessage object to the given output stream.

Inherited Member Summary

Methods inherited from class `Object`

`clone()`, `equals(Object)`, `finalize()`, `getClass()`, `hashCode()`, `notify()`, `notifyAll()`, `toString()`, `wait(long, int)`, `wait(long, int)`, `wait(long, int)`

Fields

`CHARACTER_SET_ENCODING`

```
public static final java.lang.String CHARACTER_SET_ENCODING
```

Specifies the character type encoding for the SOAP Message. Valid values include “utf-8” and “utf-16”. See vendor documentation for additional supported values. The default is “utf-8”.

Since: SAAJ 1.2

See Also: [SOAPMessage.setProperty₁₃₇](#)

`WRITE_XML_DECLARATION`

```
public static final java.lang.String WRITE_XML_DECLARATION
```

Specifies whether the SOAP Message will contain an XML declaration when it is sent. The only valid values are “true” and “false”. The default is “false”.

Since: SAAJ 1.2

See Also: [SOAPMessage.setProperty₁₃₇](#)

Constructors

`SOAPMessage()`

```
public SOAPMessage()
```

Methods

`addAttachmentPart(AttachmentPart)`

```
public abstract void addAttachmentPart(javax.xml.soap.AttachmentPart5
    attachmentPart)
```

Adds the given `AttachmentPart` object to this `SOAPMessage` object. An `AttachmentPart` object must be created before it can be added to a message.

Parameters:

`attachmentPart` - an `AttachmentPart` object that is to become part of this `SOAPMessage` object

Throws:

`java.lang.IllegalArgumentException` - if there was a problem with the specified `attachmentPart` object

countAttachments()

```
public abstract int countAttachments()
```

Gets a count of the number of attachments in this message. This count does not include the SOAP part.

Returns: the number of `AttachmentPart` objects that are part of this `SOAPMessage` object

createAttachmentPart()

```
public abstract javax.xml.soap.AttachmentPart createAttachmentPart()
```

Creates a new empty `AttachmentPart` object. Note that the method `addAttachmentPart` must be called with this new `AttachmentPart` object as the parameter in order for it to become an attachment to this `SOAPMessage` object.

Returns: a new `AttachmentPart` object that can be populated and added to this `SOAPMessage` object

createAttachmentPart(DataHandler)

```
public javax.xml.soap.AttachmentPart createAttachmentPart(DataHandler  
dataHandler)
```

Creates an `AttachmentPart` object and populates it using the given `DataHandler` object.

Parameters:

`dataHandler` - the `javax.activation.DataHandler` object that will generate the content for this `SOAPMessage` object

Returns: a new `AttachmentPart` object that contains data generated by the given `DataHandler` object

Throws:

`java.lang.IllegalArgumentException` - if there was a problem with the specified `DataHandler` object

See Also: `javax.activation.DataHandler`,
`javax.activation.DataContentHandler`

createAttachmentPart(Object, String)

```
public javax.xml.soap.AttachmentPart5 createAttachmentPart(java.lang.Object  
content, java.lang.String contentType)
```

Creates an AttachmentPart object and populates it with the specified data of the specified content type. The type of the Object should correspond to the value given for the Content-Type.

Parameters:

content - an Object containing the content for the AttachmentPart object to be created
contentType - a String object giving the type of content; examples are “text/xml”, “text/plain”, and “image/jpeg”

Returns: a new AttachmentPart object that contains the given data

Throws:

`java.lang.IllegalArgumentException` - may be thrown if the contentType does not match the type of the content object, or if there was no DataContentHandler object for the given content object

See Also: `javax.activation.DataHandler`,
`javax.activation.DataContentHandler`

getAttachment(SOAPElement)

```
public abstract javax.xml.soap.AttachmentPart5  
getAttachment(javax.xml.soap.SOAPElement67 element)  
throws SOAPException
```

Returns an AttachmentPart object that is associated with an attachment that is referenced by this SOAPElement or null if no such attachment exists. References can be made via an href attribute as described in SOAP Messages with Attachments (<http://www.w3.org/TR/SOAP-attachments#SOAPReferenceToAttachments>) , or via a single Text child node containing a URI as described in the WS-I Attachments Profile 1.0 for elements of schema type *ref:swaRef* (*ref:swaRef* (<http://www.w3.org/Profiles/AttachmentsProfile-1.0-2004-08-24.html>)). These two mechanisms must be supported. The support for references via href attribute also implies that this method should also be supported on an element that is an *xop:Include* element (*XOP* (<http://www.w3.org/2000/xp/Group/3/06/Attachments/XOP.html>)). other reference mechanisms may be supported by individual implementations of this standard. Contact your vendor for details.

Parameters:

element - The SOAPElement containing the reference to an Attachment

Returns: the referenced AttachmentPart or null if no such AttachmentPart exists or no reference can be found in this SOAPElement.

Throws:

SOAPException₈₉ - if there is an error in the attempt to access the attachment

Since: SAAJ 1.3

getAttachments()

```
public abstract java.util.Iterator<AttachmentPart> getAttachments()
```

Retrieves all the AttachmentPart objects that are part of this SOAPMessage object.

Returns: an iterator over all the attachments in this message

getAttachments(MimeHeaders)

```
public abstract java.util.Iterator<AttachmentPart>  
    getAttachments(javax.xml.soap.MimeHeaders28 headers)
```

Retrieves all the AttachmentPart objects that have header entries that match the specified headers. Note that a returned attachment could have headers in addition to those specified.

Parameters:

headers - a MimeHeaders object containing the MIME headers for which to search

Returns: an iterator over all attachments that have a header that matches one of the given headers

getContentDescription()

```
public abstract java.lang.String getContentDescription()
```

Retrieves a description of this SOAPMessage object's content.

Returns: a String describing the content of this message or null if no description has been set

See Also: setContentDescription(String)₁₃₇

getMimeHeaders()

```
public abstract javax.xml.soap.MimeHeaders28 getMimeHeaders()
```

Returns all the transport-specific MIME headers for this SOAPMessage object in a transport-independent fashion.

Returns: a MimeHeaders object containing the MimeHeader objects

getProperty(String)

```
public java.lang.Object getProperty(java.lang.String property)
    throws SOAPException
```

Retrieves value of the specified property.

Parameters:

property - the name of the property to retrieve

Returns: the value associated with the named property or null if no such property exists.

Throws:

[SOAPException₈₉](#) - if the property name is not recognized.

Since: SAAJ 1.2

getSOAPBody()

```
public javax.xml.soap.SOAPBody46 getSOAPBody()
    throws SOAPException
```

Gets the SOAP Body contained in this SOAPMessage object.

Returns: the SOAPBody object contained by this SOAPMessage object

Throws:

[SOAPException₈₉](#) - if the SOAP Body does not exist or cannot be retrieved

Since: SAAJ 1.2

getSOAPHeader()

```
public javax.xml.soap.SOAPHeader115 getSOAPHeader()
    throws SOAPException
```

Gets the SOAP Header contained in this SOAPMessage object.

Returns: the SOAPHeader object contained by this SOAPMessage object

Throws:

[SOAPException₈₉](#) - if the SOAP Header does not exist or cannot be retrieved

Since: SAAJ 1.2

getSOAPPart()

```
public abstract javax.xml.soap.SOAPPart139 getSOAPPart()
```

Gets the SOAP part of this SOAPMessage object.

SOAPMessage object contains one or more attachments, the SOAP Part must be the first MIME body part in the message.

Returns: the SOAPPart object for this SOAPMessage object

removeAllAttachments()

```
public abstract void removeAllAttachments()
```

Removes all AttachmentPart objects that have been added to this SOAPMessage object.

This method does not touch the SOAP part.

removeAttachments(MimeHeaders)

```
public abstract void removeAttachments(javax.xml.soap.MimeHeaders28 headers)
```

Removes all the AttachmentPart objects that have header entries that match the specified headers. Note that the removed attachment could have headers in addition to those specified.

Parameters:

headers - a MimeHeaders object containing the MIME headers for which to search

Since: SAAJ 1.3

saveChanges()

```
public abstract void saveChanges()  
    throws SOAPException
```

Updates this SOAPMessage object with all the changes that have been made to it. This method is called automatically when [writeTo\(OutputStream\)₁₃₈](#) is called. However, if changes are made to a message that was received or to one that has already been sent, the method saveChanges needs to be called explicitly in order to save the changes. The method saveChanges also generates any changes that can be read back (for example, a MessageId in profiles that support a message id). All MIME headers in a message that is created for sending purposes are guaranteed to have valid values only after saveChanges has been called.

In addition, this method marks the point at which the data from all constituent AttachmentPart objects are pulled into the message.

Throws:

`SOAPException` - if there was a problem saving changes to this message.

[SOAPException₈₉](#)

saveRequired()

```
public abstract boolean saveRequired()
```

Indicates whether this SOAPMessage object needs to have the method `saveChanges` called on it.

Returns: `true` if `saveChanges` needs to be called; `false` otherwise.

setContentDescription(String)

```
public abstract void setContentDescription(java.lang.String description)
```

Sets the description of this SOAPMessage object's content with the given description.

Parameters:

`description` - a String describing the content of this message

See Also: [getContentDescription\(\)](#) ¹³⁴

setProperty(String, Object)

```
public void setProperty(java.lang.String property, java.lang.Object value)  
    throws SOAPException
```

Associates the specified value with the specified property. If there was already a value associated with this property, the old value is replaced.

The valid property names include [WRITE_XML_DECLARATION](#) ¹³¹ and [CHARACTER_SET_ENCODING](#) ¹³¹. All of these standard SAAJ properties are prefixed by "javax.xml.soap". Vendors may also add implementation specific properties. These properties must be prefixed with package names that are unique to the vendor.

Setting the property `WRITE_XML_DECLARATION` to "true" will cause an XML Declaration to be written out at the start of the SOAP message. The default value of "false" suppresses this declaration.

The property `CHARACTER_SET_ENCODING` defaults to the value "utf-8" which causes the SOAP message to be encoded using UTF-8. Setting `CHARACTER_SET_ENCODING` to "utf-16" causes the SOAP message to be encoded using UTF-16.

Some implementations may allow encodings in addition to UTF-8 and UTF-16. Refer to your vendor's documentation for details.

Parameters:

`property` - the property with which the specified value is to be associated.

`value` - the value to be associated with the specified property

Throws:

[SOAPException](#) ₈₉ - if the property name is not recognized.

Since: SAAJ 1.2

writeTo(OutputStream)

```
public abstract void writeTo(java.io.OutputStream out)  
    throws SOAPException, IOException
```

Writes this SOAPMessage object to the given output stream. The externalization format is as defined by the SOAP 1.1 with Attachments specification.

If there are no attachments, just an XML stream is written out. For those messages that have attachments, writeTo writes a MIME-encoded byte stream.

Note that this method does not write the transport-specific MIME Headers of the Message

Parameters:

out - the OutputStream object to which this SOAPMessage object will be written

Throws:

java.io.IOException - if an I/O error occurs

SOAPException_{g9} - if there was a problem in externalizing this SOAP message

2.27 SOAPPart

Declaration

```
public abstract class SOAPPart implements org.w3c.dom.Document, javax.xml.soap.Node35
```

```
java.lang.Object
|
+-- javax.xml.soap.SOAPPart
```

All Implemented Interfaces: org.w3c.dom.Document, org.w3c.dom.Node, javax.xml.soap.Node₃₅

Description

The container for the SOAP-specific portion of a SOAPMessage object. All messages are required to have a SOAP part, so when a SOAPMessage object is created, it will automatically have a SOAPPart object.

A SOAPPart object is a MIME part and has the MIME headers Content-Id, Content-Location, and Content-Type. Because the value of Content-Type must be “text/xml”, a SOAPPart object automatically has a MIME header of Content-Type with its value set to “text/xml”. The value must be “text/xml” because content in the SOAP part of a message must be in XML format. Content that is not of type “text/xml” must be in an AttachmentPart object rather than in the SOAPPart object.

When a message is sent, its SOAP part must have the MIME header Content-Type set to “text/xml”. Or, from the other perspective, the SOAP part of any message that is received must have the MIME header Content-Type with a value of “text/xml”.

A client can access the SOAPPart object of a SOAPMessage object by calling the method SOAPMessage.getSOAPPart. The following line of code, in which message is a SOAPMessage object, retrieves the SOAP part of a message.

```
SOAPPart soapPart = message.getSOAPPart();
```

A SOAPPart object contains a SOAPEnvelope object, which in turn contains a SOAPBody object and a SOAPHeader object. The SOAPPart method getEnvelope can be used to retrieve the SOAPEnvelope object.

Member Summary

Constructors

[SOAPPart\(\)](#) ¹⁴¹

Methods

abstract void	addMimeHeader(java.lang.String name, java.lang.String value) ¹⁴²	Creates a MimeHeader object with the specified name and value and adds it to this SOAPPart object.
abstract	getAllMimeHeaders() ¹⁴²	
java.util.Iterator<MimeHeader>		Retrieves all the headers for this SOAPPart object as an iterator over the MimeHeader objects.
abstract	getContent() ¹⁴²	
javax.xml.transform.Source		Returns the content of the SOAPEnvelope as a JAXP Source object.
java.lang.String	getContentId() ¹⁴²	Retrieves the value of the MIME header whose name is “Content-Id”.
java.lang.String	getContentLocation() ¹⁴³	Retrieves the value of the MIME header whose name is “Content-Location”.
abstract SOAPEnvelope	getEnvelope() ¹⁴³	Gets the SOAPEnvelope object associated with this SOAPPart object.
abstract	getMatchingMimeHeaders(java.lang.String[] names) ¹⁴³	
java.util.Iterator<MimeHeader>		Retrieves all MimeHeader objects that match a name in the given array.
abstract	getMimeHeader(java.lang.String name) ¹⁴³	
java.lang.String[]		Gets all the values of the MimeHeader object in this SOAPPart object that is identified by the given String.
abstract	getNonMatchingMimeHeaders(java.lang.String[] names) ¹⁴⁴	
java.util.Iterator<MimeHeader>		Retrieves all MimeHeader objects whose name does not match a name in the given array.
abstract void	removeAllMimeHeaders() ¹⁴⁴	Removes all the MimeHeader objects for this SOAPEnvelope object.
abstract void	removeMimeHeader(java.lang.String header) ¹⁴⁴	Removes all MIME headers that match the given name.
abstract void	setContent(javax.xml.transform.Source source) ¹⁴⁴	Sets the content of the SOAPEnvelope object with the data from the given Source object.
void	setContentId(java.lang.String contentId) ¹⁴⁴	Sets the value of the MIME header named “Content-Id” to the given String.
void	setContentLocation(java.lang.String contentLocation) ¹⁴⁵	Sets the value of the MIME header “Content-Location” to the given String.
abstract void	setMimeHeader(java.lang.String name, java.lang.String value) ¹⁴⁵	Changes the first header entry that matches the given header name so that its value is the given value, adding a new header with the given name and value if no existing header is a match.

Inherited Member Summary

Fields inherited from interface Node

ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE

Methods inherited from interface Document

createAttribute(String), createAttributeNS(String, String),
createCDATASection(String), createComment(String), createDocumentFragment(),
createElement(String), createElementNS(String, String),
createEntityReference(String), createProcessingInstruction(String, String),
createTextNode(String), getDoctype(), getDocumentElement(), getElementById(String),
getElementsByTagName(String), getElementsByTagNameNS(String, String),
getImplementation(), importNode(Node, boolean)

Methods inherited from interface Node

appendChild(Node), cloneNode(boolean), getAttributes(), getChildNodes(),
getFirstChild(), getLastChild(), getLocalName(), getNamespaceURI(), getNextSibling(),
getNodeName(), getNodeType(), getNodeValue(), getOwnerDocument(), getParentNode(),
getPrefix(), getPreviousSibling(), hasAttributes(), hasChildNodes(),
insertBefore(Node, Node), isSupported(String, String), normalize(),
removeChild(Node), replaceChild(Node, Node), setNodeValue(String), setPrefix(String)

Methods inherited from interface Node₃₅

detachNode()₃₆, getParentElement()₃₆, getValue()₃₆, recycleNode()₃₇,
setParentElement(SOAPElement)₃₇, setValue(String)₃₇

Methods inherited from class Object

clone(), equals(Object), finalize(), getClass(), hashCode(), notify(), notifyAll(),
toString(), wait(long, int), wait(long, int), wait(long, int)

Constructors

SOAPPart()

```
public SOAPPart()
```

Methods

addMimeHeader(String, String)

```
public abstract void addMimeHeader(java.lang.String name, java.lang.String value)
```

Creates a `MimeHeader` object with the specified name and value and adds it to this `SOAPPart` object. If a `MimeHeader` with the specified name already exists, this method adds the specified value to the already existing value(s).

Note that RFC822 headers can contain only US-ASCII characters.

Parameters:

name - a `String` giving the header name

value - a `String` giving the value to be set or added

Throws:

`java.lang.IllegalArgumentException` - if there was a problem with the specified mime header name or value

getAllMimeHeaders()

```
public abstract java.util.Iterator<MimeHeader> getAllMimeHeaders()
```

Retrieves all the headers for this `SOAPPart` object as an iterator over the `MimeHeader` objects.

Returns: an `Iterator` object with all of the Mime headers for this `SOAPPart` object

getContent()

```
public abstract javax.xml.transform.Source getContent()  
    throws SOAPException
```

Returns the content of the `SOAPEnvelope` as a JAXP `Source` object.

Returns: the content as a `javax.xml.transform.Source` object

Throws:

[SOAPException](#)₈₉ - if the implementation cannot convert the specified `Source` object

See Also: [setContent\(Source\)](#)₁₄₄

getContentId()

```
public java.lang.String getContentId()
```

Retrieves the value of the MIME header whose name is “Content-Id”.

Returns: a `String` giving the value of the MIME header named “Content-Id”

See Also: [setContentId\(String\)](#)₁₄₄

getContentLocation()

```
public java.lang.String getContentLocation()
```

Retrieves the value of the MIME header whose name is “Content-Location”.

Returns: a `String` giving the value of the MIME header whose name is “Content-Location”

See Also: [setContentLocation\(String\)](#) ₁₄₅

getEnvelope()

```
public abstract javax.xml.soap.SOAPEnvelope84 getEnvelope()  
    throws SOAPException
```

Gets the `SOAPEnvelope` object associated with this `SOAPPart` object. Once the SOAP envelope is obtained, it can be used to get its contents.

Returns: the `SOAPEnvelope` object for this `SOAPPart` object

Throws:

[SOAPException](#)₈₉ - if there is a SOAP error

getMatchingMimeHeaders(String[])

```
public abstract java.util.Iterator<MimeHeader>  
    getMatchingMimeHeaders(java.lang.String[] names)
```

Retrieves all `MimeHeader` objects that match a name in the given array.

Parameters:

names - a `String` array with the name(s) of the MIME headers to be returned

Returns: all of the MIME headers that match one of the names in the given array, returned as an `Iterator` object

getMimeHeader(String)

```
public abstract java.lang.String[] getMimeHeader(java.lang.String name)
```

Gets all the values of the `MimeHeader` object in this `SOAPPart` object that is identified by the given `String`.

Parameters:

name - the name of the header; example: “Content-Type”

Returns: a `String` array giving all the values for the specified header

See Also: [setMimeHeader\(String, String\)](#) ₁₄₅

getNonMatchingMimeHeaders(String[])

```
public abstract java.util.Iterator<MimeHeader>  
    getNonMatchingMimeHeaders(java.lang.String[] names)
```

Retrieves all `MimeHeader` objects whose name does not match a name in the given array.

Parameters:

names - a `String` array with the name(s) of the MIME headers not to be returned

Returns: all of the MIME headers in this `SOAPPart` object except those that match one of the names in the given array. The nonmatching MIME headers are returned as an `Iterator` object.

removeAllMimeHeaders()

```
public abstract void removeAllMimeHeaders()
```

Removes all the `MimeHeader` objects for this `SOAPEnvelope` object.

removeMimeHeader(String)

```
public abstract void removeMimeHeader(java.lang.String header)
```

Removes all MIME headers that match the given name.

Parameters:

header - a `String` giving the name of the MIME header(s) to be removed

setContent(Source)

```
public abstract void setContent(javax.xml.transform.Source source)  
    throws SOAPException
```

Sets the content of the `SOAPEnvelope` object with the data from the given `Source` object. This `Source` must contain a valid SOAP document.

Parameters:

source - the `javax.xml.transform.Source` object with the data to be set

Throws:

[SOAPException](#)₈₉ - if there is a problem in setting the source

See Also: [getContent\(\)](#)₁₄₂

setContentId(String)

```
public void setContentId(java.lang.String contentId)
```

Sets the value of the MIME header named “Content-Id” to the given `String`.

Parameters:

`contentId` - a `String` giving the value of the MIME header “Content-Id”

Throws:

`java.lang.IllegalArgumentException` - if there is a problem in setting the content id

See Also: [getContentId\(\)](#) ₁₄₂

setContentLocation(String)

```
public void setContentLocation(java.lang.String contentLocation)
```

Sets the value of the MIME header “Content-Location” to the given `String`.

Parameters:

`contentLocation` - a `String` giving the value of the MIME header “Content-Location”

Throws:

`java.lang.IllegalArgumentException` - if there is a problem in setting the content location.

See Also: [getContentLocation\(\)](#) ₁₄₃

setMimeHeader(String, String)

```
public abstract void setMimeHeader(java.lang.String name, java.lang.String value)
```

Changes the first header entry that matches the given header name so that its value is the given value, adding a new header with the given name and value if no existing header is a match. If there is a match, this method clears all existing values for the first header that matches and sets the given value instead. If more than one header has the given name, this method removes all of the matching headers after the first one.

Note that RFC822 headers can contain only US-ASCII characters.

Parameters:

`name` - a `String` giving the header name for which to search

`value` - a `String` giving the value to be set. This value will be substituted for the current value(s) of the first header that is a match if there is one. If there is no match, this value will be the value for a new `MimeHeader` object.

Throws:

`java.lang.IllegalArgumentException` - if there was a problem with the specified mime header name or value

See Also: [getMimeHeader\(String\)](#) ₁₄₃

2.28 Text

Declaration

```
public interface Text extends Node35, org.w3c.dom.Text
```

All Superinterfaces: org.w3c.dom.CharacterData, org.w3c.dom.Node, Node₃₅, org.w3c.dom.Text

Description

A representation of a node whose value is text. A Text object may represent text that is content or text that is a comment.

Member Summary	
Methods	
boolean	isComment() ₁₄₇ Retrieves whether this Text object represents a comment.

Inherited Member Summary	
Fields inherited from interface Node	
ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE, ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE	
Methods inherited from interface CharacterData	
appendData(String), deleteData(int, int), getData(), getLength(), insertData(int, String), replaceData(int, int, String), setData(String), substringData(int, int)	
Methods inherited from interface Node₃₅	
detachNode() ₃₆ , getParentElement() ₃₆ , getValue() ₃₆ , recycleNode() ₃₇ , setParentElement(SOAPElement) ₃₇ , setValue(String) ₃₇	

Inherited Member Summary

Methods inherited from interface **Node**

`appendChild(Node)`, `cloneNode(boolean)`, `getAttributes()`, `getChildNodes()`,
`getFirstChild()`, `getLastChild()`, `getLocalName()`, `getNamespaceURI()`, `getNextSibling()`,
`getNodeName()`, `getNodeType()`, `getNodeValue()`, `getOwnerDocument()`, `getParentNode()`,
`getPrefix()`, `getPreviousSibling()`, `hasAttributes()`, `hasChildNodes()`,
`insertBefore(Node, Node)`, `isSupported(String, String)`, `normalize()`,
`removeChild(Node)`, `replaceChild(Node, Node)`, `setNodeValue(String)`, `setPrefix(String)`

Methods inherited from interface **Text**

`splitText(int)`

Methods

isComment()

```
public boolean isComment()
```

Retrieves whether this `Text` object represents a comment.

Returns: `true` if this `Text` object is a comment; `false` otherwise

A

References

For more information, refer to the following web sites:

- SOAP 1.1

<http://www.w3.org/TR/SOAP>

- SOAP 1.2

<http://www.w3.org/TR/soap12-part1/>

- SOAP Messages with Attachments

<http://www.w3.org/TR/SOAP-attachments>,

<http://www.w3.org/TR/soap12-af>

- JavaBeans™ Activation Framework Version 1.0a

<http://www.oracle.com/technetwork/articles/java/index-135046.html>

- Java™ API for XML Processing Version 1.2 Final Release

<https://jcp.org/en/jsr/detail?id=63>

- Java™ API for XML Messaging Version 1.1 Final Release

<https://jcp.org/en/jsr/detail?id=67>

- WS-I Attachments Profile 1.0

<http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html>

