# The Java API for XML Based RPC (JAX-RPC) 2.0

*Editors Draft*
*February 11, 2004*

Editors:
Roberto Chinnici,
Marc Hadley
Comments to: jaxrpc-spec-comments@sun.com

*Sun Microsystems, Inc.*
*4150 Network Circle*
*Santa Clara, CA 95054 USA*

**Java(TM) API for XML based Remote Procedure Call (JAX-RPC) 2.0 Specification ("Specification")**
**Version: 2.0**
**Status: editors copy**
**Release: February 11, 2004**



**NOTICE**

The Specification is protected by copyright and the information described therein may be protected by one or more U.S. patents, foreign patents, or pending applications. Except as provided under the following license, no part of the Specification may be reproduced in any form by any means without the prior written authorization of Sun Microsystems, Inc. ("Sun") and its licensors, if any. Any use of the Specification and the information described therein will be governed by the terms and conditions of this Agreement.

Subject to the terms and conditions of this license, Sun hereby grants you a fully-paid, non-exclusive, non-transferable, limited license (without the right to sublicense) under Sun's intellectual property rights to review the Specification only for the purposes of evaluation. This license includes the right to discuss the Specification (including the right to provide limited excerpts of text to the extent relevant to the point[s] under discussion) with other licensees (under this or a substantially similar version of this Agreement) of the Specification. Other than this limited license, you acquire no right, title or interest in or to the Specification or any other Sun intellectual property, and the Specification may only be used in accordance with the license terms set forth herein. This license will expire on the earlier of: (i) two (2) years from the date of Release listed above; (ii) the date on which the final version of the Specification is publicly released; or (iii) the date on which the Java Specification Request (JSR) to which the Specification corresponds is withdrawn. In addition, this license will terminate immediately without notice from Sun if you fail to comply with any provision of this license. Upon termination, you must cease use of or destroy the Specification.

**TRADEMARKS**

No right, title, or interest in or to any trademarks, service marks, or trade names of Sun or Sun's licensors is granted hereunder. Sun, Sun Microsystems, the Sun logo, Java, and the Java Coffee Cup logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

**DISCLAIMER OF WARRANTIES**

THE SPECIFICATION IS PROVIDED "AS IS" AND IS EXPERIMENTAL AND MAY CONTAIN DEFECTS OR DEFICIENCIES WHICH CANNOT OR WILL NOT BE CORRECTED BY SUN. SUN MAKES NO REPRESENTATIONS OR WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT THAT THE CONTENTS OF THE SPECIFICATION ARE SUITABLE FOR ANY PURPOSE OR THAT ANY PRACTICE OR IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER RIGHTS. This document does not represent any commitment to release or implement any portion of the Specification in any product.

THE SPECIFICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION THEREIN; THESE CHANGES WILL BE INCORPORATED INTO NEW VERSIONS OF THE SPECIFICATION, IF ANY. SUN MAY MAKE IMPROVEMENTS AND/OR CHANGES TO THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THE SPECIFICATION AT ANY TIME. Any use of such changes in the Specification will be governed by the then-current license for the applicable version of the Specification.

**LIMITATION OF LIABILITY**

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SUN 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 TO ANY FURNISHING, PRACTICING, MODIFYING OR ANY USE OF THE SPECIFICATION, EVEN IF SUN AND/OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You will hold Sun (and its licensors) harmless from any claims based on your use of the Specification for any purposes other than the limited right of evaluation as described above, and from 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**

If this Software 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**

You may wish to report any ambiguities, inconsistencies or inaccuracies you may find in connection with your evaluation of the Specification ("Feedback"). To the extent that you provide Sun with any Feedback, you hereby: (i) agree that such Feedback is provided on a non-proprietary and non-confidential basis, and (ii) grant Sun 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 related to the Specification and future versions, implementations, and test suites thereof.

**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.

Neither party may assign or otherwise transfer any of its rights or obligations under this Agreement, without the prior written consent of the other party, except that Sun may assign this Agreement to an affiliated company.

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.

*(LFI#126151/Form ID#011801)*

# Contents

# Chapter 1

# Introduction

A remote procedure call (RPC) mechanism allows a client to invoke the methods of a remote service using a familiar local procedure call paradigm. On the client, the RPC infrastructure manages the task of converting the local procedure call arguments into some standard request representation, communicating the request to the remote service and converting any response back into procedure call return values. On the server, the RPC infrastructure manages the task of converting incoming requests into local procedure calls, converting the result of local procedure calls into responses and communicating responses to the client.

XML[1] is a platform-independent means of representing structured information. XML based RPC mechanisms use XML for the representation of RPC requests and responses and inherit XML's platform independence. SOAP[2, 3, 4] describes one such XML based RPC mechanism and "defines, using XML technologies, an extensible messaging framework containing a message construct that can be exchanged over a variety of underlying protocols".

WSDL[5] is "an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information". WSDL can be considered the defacto interface definition language(IDL) for XML based RPC.

JAX-RPC 1.0[6] defines APIs and conventions for supporting XML based RPC in the Java™ platform. JAX-RPC 1.1[7] adds support for the WS-I Basic Profile 1.0[8] to improve interoperability between JAX-RPC implementations and with services implemented using other technologies.

JAX-RPC 2.0 (this specification) supersedes JAX-RPC 1.1, extending it as described in the following sections.

## 1.1 Goals

Since the release of JAX-RPC 1.0[6], new specifications and new versions of the standards it depends on have been released. JAX-RPC 2.0 relates to these specifications and standards as follows:

**JAXB** Due primarily to scheduling concerns, JAX-RPC 1.0 defined its own data binding facilities. With the release of JAXB 1.0[9] there is no reason to maintain two separate sets of XML mapping rules in the Java™ platform. JAX-RPC 2.0 will delegate data binding-related tasks to the JAXB 2.0[10] specification that is being developed in parallel with JAX-RPC 2.0.

JAXB 2.0[10] will add support for Java to XML mapping, additional support for less used XML schema constructs and provide bidirectional customization of Java $\Leftrightarrow$ XML data binding. JAX-

RPC 2.0 will allow full use of JAXB provided facilities including binding customization and optional schema validation.

**SOAP 1.2** Whilst SOAP 1.1 is still widely deployed, it's expected that services will migrate to SOAP 1.2[3, 4] now that it is a W3C Recommendation. JAX-RPC 2.0 will add support for SOAP 1.2 whilst requiring continued support for SOAP 1.1.

**WSDL 1.2** The W3C is expected to progress WSDL 1.2[11] to Recommendation during the lifetime of this JSR. JAX-RPC 2.0 will add support for WSDL 1.2 whilst requiring continued support for WSDL 1.1.

**WS-I Basic Profile 1.1** JAX-RPC 1.1 added support for WS-I Basic Profile 1.0. WS-I Basic Profile 1.1 is expected to supersede 1.0 during the lifetime of this JSR and JAX-RPC 2.0 will add support for the additional clarifications it provides.

**A Metadata Facility for the Java Programming Language (JSR 175)** JAX-RPC 2.0 will use Java annotations[12] to simplify the most common development scenarios for both clients and servers.

**Web Services Metadata for the Java™ Platform (JSR 181)** JAX-RPC 2.0 will align with and complement the annotations defined by JSR 181[13].

**Implementing Enterprise Web Services (JSR 109)** The JSR 109[14] defined `jaxrpc-mapping-info` deployment descriptor provides deployment time Java ⇔ WSDL mapping functionality. In conjunction with JSR 181[13], JAX-RPC 2.0 will complement this mapping functionality with development time Java annotations that control Java ⇔ WSDL mapping.

**Web Services Security (JSR 183)** JAX-RPC 2.0 will align with and complement the security APIs defined by JSR 183[15].

JAX-RPC 2.0 will improve support for document/message centric usage:

**Asynchrony** JAX-RPC 2.0 will add support for client side asynchronous operations.

**Non-HTTP Transports** JAX-RPC 2.0 will improve the separation between the XML based RPC framework and the underlying transport mechanism to simplify use of JAX-RPC with non-HTTP transports.

**Message Access** JAX-RPC 2.0 will simplify client and service access to the messages underlying an exchange.

**Session Management** JAX-RPC 1.1 session management capabilities are tied to HTTP. JAX-RPC 2.0 will add support for message based session management.

JAX-RPC 2.0 will also address issues that have arisen with experience of implementing and using JAX-RPC 1.0:

**Inclusion in J2SE** JAX-RPC 2.0 will prepare JAX-RPC for inclusion in a future version of J2SE. Application portability is a key requirement and JAX-RPC 2.0 will define mechanisms to produce fully portable clients.

**Handlers** JAX-RPC 2.0 will simplify the development of handlers and will provide a mechanism to allow handlers to collaborate with service clients and service endpoint implementations.

**Versioning and Evolution of Web Services** JAX-RPC 2.0 will describe techniques and mechanisms to ease the burden on developers when creating new versions of existing services.

**Backwards Compatibility of Binary Artifacts** JAX-RPC 2.0 will not preclude preservation of binary compatibility between JAX-RPC 1.x and 2.0 implementation runtimes.

## 1.2 Non-Goals

The following are non-goals:

**Pluggable data binding** JAX-RPC 2.0 will defer data binding to JAXB[10], it is not a goal to provide a plug-in API to allow other types of data binding technologies to be used in place of JAXB. However, JAX-RPC 2.0 will maintain the capability to selectively disable data binding to provide an XML based fragment suitable for use as input to alternative data binding technologies.

**SOAP Encoding Support** Use of the SOAP encoding is essentially deprecated in the web services community, e.g. the WS-I Basic Profile[8] excludes SOAP encoding. Instead, literal usage is preferred, either in the RPC or document style.

SOAP 1.1 encoding is supported in JAX-RPC 1.0 and 1.1 but its support in JAX-RPC 2.0 runs counter to the goal of delegation of data binding to JAXB. Therefore JAX-RPC 2.0 will make support for SOAP 1.1 encoding optional and defer description of it to JAX-RPC 1.1.

Support for the SOAP 1.2 Encoding[4] is optional in SOAP 1.2 and JAX-RPC 2.0 will not add support for SOAP 1.2 encoding.

**Backwards Compatibility of Generated Artifacts** JAX-RPC 1.0 and JAXB 1.0 bind XML to Java in different ways. Generating source code that works with unmodified JAX-RPC 1.x client source code is not a goal.

**Support for Java versions prior to J2SE 1.5** JAX-RPC 2.0 relies on many of the Java language features added in J2SE 1.5. It is not a goal to support JAX-RPC 2.0 on Java versions prior to J2SE 1.5.

**Service Registration and Discovery** It is not a goal of JAX-RPC 2.0 to describe registration and discovery of services via UDDI or ebXML RR. This capability is provided independently by JAXR[16].

## 1.3 Requirements

### 1.3.1 Describe Relationship To JAXB

JAX-RPC describes the WSDL ⇔ Java mapping, but data binding is delegated to JAXB[10]. The specification must clearly designate where JAXB rules apply to the WSDL ⇔ Java mapping without reproducing those rules and must describe how JAXB capabilities (e.g. the JAXB binding language) are incorporated into JAX-RPC. JAX-RPC is required to be able to influence the JAXB binding, e.g. to avoid name collisions and to be able to control schema validation on serialization and deserialization.

### 1.3.2 Standardized WSDL Mapping

WSDL is the de-facto interface definition language for XML-based RPC. The specification must specify a standard WSDL ⇔ Java mapping. The following versions of WSDL must be supported:

- WSDL 1.1[5] as clarified by the WS-I Basic Profile[8, 17],
- WSDL 1.2[11, 18, 19].

The standardized WSDL mapping will describe the default WSDL ⇔ Java mapping. The default mapping may be overridden using customizations as described below.

### 1.3.3 Customizable WSDL Mapping

The specification must provide a standard way to customize the WSDL ⇔ Java mapping. The following customization methods will be specified:

**Java Annotations** The specification will define a set of standard annotations that may be used in Java source files to specify the mapping from Java artifacts to their associated WSDL components. The annotations will support mapping to both WSDL 1.1 and WSDL 1.2. The annotations defined by JAX-RPC will mesh cleanly with those defined by JAXB[10] and JSR 181[13].

**WSDL Annotations** The specification will define a set of standard annotations that may be used either within WSDL documents or as in an external form to specify the mapping from WSDL components to their associated Java artifacts. The annotations will support mapping from both WSDL 1.1 and WSDL 1.2. The annotations defined by JAX-RPC will mesh cleanly with those defined by the JAXB binding language.

The specification must describe the precedence rules governing combinations of the customization methods.

### 1.3.4 Standardized Protocol Bindings

The specification must describe standard bindings to the following protocols:

- SOAP 1.1[2] as clarified by the WS-I Basic Profile[8, 17],
- SOAP 1.2[3, 4].

The specification must not prevent non-standard bindings to other protocols.

### 1.3.5 Standardized Transport Bindings

The specification must describe standard bindings to the following protocols:

- HTTP/1.1[20].

The specification must not prevent non-standard bindings to other transports.

### 1.3.6 Standardized Handler Framework

The specification must include a standardized handler framework that describes:

**Data binding for handlers** The framework will offer data binding facilities to handlers and will support handlers that are decoupled from the SAAJ API.

**Handler Context** The framework will describe a mechanism for communicating properties between handlers and the associated service clients and service endpoint implementations.

**Bidirectional handler chains** Support for bidirectional handler chains that are used for both outgoing and incoming messages will be added to the existing unidirectional handler chains.

**Unified Response and Fault Handling** The `handleResponse` and `handleFault` methods will be unified and the the declarative model for handlers will be improved.

### 1.3.7 Versioning and Evolution

The specification must describe techniques and mechanisms to support versioning of service endpoint interfaces. The facilities must allow new versions of an interface to be deployed whilst maintaining compatibility for existing clients.

### 1.3.8 Standardized Synchronous and Asynchronous Invocation

There must be a detailed description of the generated method signatures to support both asynchronous and synchronous method invocation in stubs generated by JAX-RPC. Both forms of invocation will support a user configurable timeout period.

### 1.3.9 Session Management

The specification must describe a standard session management mechanism including:

**Session APIs** Definition of a session interface and methods to obtain the session interface and initiate sessions for handlers and service endpoint implementations.

**HTTP based sessions** The session management mechanism must support HTTP cookies and URL rewriting.

**SOAP based sessions** There must be a standardized way to identify the location of a session identifier in SOAP message content using Java or WSDL annotations.

## 1.4 Use Cases

### 1.4.1 Handler Framework

#### Reliable Messaging Support

A developer wishes to add support for a reliable messaging SOAP feature to an existing service endpoint. The support takes the form of a JAX-RPC handler.

#### Message Logging

A developer wishes to log incoming and outgoing messages for later analysis, e.g. checking messages using the WS-I testing tools.

#### WS-I Conformance Checking

A developer wishes to check incoming and outgoing messages for conformance to one or more WS-I profiles at runtime.

## 1.5 Conventions

The keywords ‘MUST’, ‘MUST NOT’, ‘REQUIRED’, ‘SHALL’, ‘SHALL NOT’, ‘SHOULD’, ‘SHOULD NOT’, ‘RECOMMENDED’, ‘MAY’, and ‘OPTIONAL’ in this document are to be interpreted as described in RFC 2119[21].

For convenience, conformance requirements are numbered and shown as follows:

**R 1.1 (Example)** Implementations MUST do something.

Java code and XML fragments are formatted as shown in figure 1.1:

Figure 1.1: Example Java Code

```
1  package com.example.hello;
2
3  public class Hello {
4      public static void main(String args[]) {
5          System.out.println("Hello World");
6      }
7  }
```

This specification uses a number of namespace prefixes throughout; they are listed in Table 1.1. Note that the choice of any namespace prefix is arbitrary and not semantically significant (see XML Infoset[22]).

| Prefix | Namespace | Notes |
|---|---|---|
| env | http://www.w3.org/2003/05/soap-envelope | A normative XML Schema[23, 24] document for the http://www.w3.org/2003/05/soap-envelope namespace can be found at http://www.w3.org/2003/05/soap-envelope. |
| xsd | http://www.w3.org/2001/XMLSchema | The namespace of the XML Schema[23, 24] specification |
| jaxb | http://java.sun.com/xml/ns/jaxb | The namespace of the JAXB [9] specification |

Table 1.1: Prefixes and Namespaces used in this specification.

Namespace names of the general form ‘http://example.org/...’ and ‘http://example.com/...’ represent application or context-dependent URIs (see RFC 2396[20]).

All parts of this specification are normative, with the exception of examples and sections explicitly marked as ‘Non-Normative’.

## 1.6 Expert Group Members

TBD

## 1.7 Acknowledgements

TBD

# Chapter 2

# Service Client APIs

This chapter describes the standard APIs provided for client side use of JAX-RPC. These APIs allow a client to configure generated stubs, create dynamic proxies for remote service endpoints and dynamically construct operation invocations.

## 2.1 javax.xml.rpc.ServiceFactory

## 2.2 javax.xml.rpc.Service

## 2.3 javax.xml.rpc.Call

**Editors Note 2.1** *The EG has agreed to reposition the Call API as a SOAP Encoding centric API and is considering whether to deprecate it in favor of the a new alternative that is more document/rpc literal friendly.*

The `Call` interface provides support for dynamically constructing operation invocations either with or without a WSDL description of the service endpoint.

**R 2.1 (Call support)** Implementations MUST support the `javax.xml.rpc.Call` interface.

### 2.3.1 Operation Configuration

`Call` instances are obtained using one of the `createCall` methods of a `Service` instance. A `Call` instance can be created in one of three states depending on the combination of how the `Service` instance was obtained and which `createCall` variant was used:

**Unconfigured** The client is responsible for configuring both the binding information and the operation information

**Binding** The binding information is configured but the client is responsible for configuring the operation information

**Configured** The binding and operation information are both configured

| | **`Service.createCall` Arguments** | | | |
|---|---|---|---|---|
| | None | `QName port` | `QName port, QName op` | `QName port, String op` |
| **`createService` Arguments** | | | | |
| `QName svc` | U | U | U | U |
| `URL wsdlDoc, QName svc` | U | B | C | C |
| **`loadService` Arguments** | | | | |
| `Class sei` | U | B? | C? | C? |
| `URL wsdlDoc, Class sei, Properties p` | U | B? | C? | C? |
| `URL wsdlDoc, QName svc, Properties p` | U | B | C | C |

Table 2.1: Call states resulting from combinations of `Service` creation and `createCall` variants.

Table 2.1 shows the state for each combination.

A client can determine whether a `Call` instance is pre-configured or not by use of the `isParameterAndReturnSpecRequired` method. This method returns `true` for pre-configured instances ('C' in table 2.1), `false` otherwise.

Unconfigured or partially configured (Binding only) `Call` instances require configuration using the appropriate setter methods prior to use of the `invoke` and `invokeOneWay` methods. `Call` instances are mutable, a client may change the configuration of an existing instance and re-use it.

**R 2.2 (Use of unconfigured `Call`)** An implementation MUST throw a `JAXRPCException` if the `invoke` or `invokeOneWay` methods are called on an unconfigured instance.

Setter methods are provided to configure:

- The name of the operation to invoke
- The names, types and modes of the operation parameters
- The operation return type
- The name of the port type
- The endpoint address
- Binding specific properties (see section 2.4 on page 9 for a list of standard properties).

**R 2.3 (Invalid `Call` configuration)** An implementation MUST throw a `JAXRPCException` if a client attempts to set an unknown or unsupported optional property or if an implementation detects an error in the value of a property.

### 2.3.2 Operation Invocation

When an operation is invoked, the `Call` instance checks that the passed parameters match the number, order and types of parameters configured in the instance.

**R 2.4 (Misconfigured invocation)** When an operation is invoked, an implementation MUST throw a `JAXRPCException` if the `Call` instance is incorrectly configured or if the passed parameters do not match the configuration.

A `Call` instance supports two invocation modes:

**Synchronous request response (`invoke` method)** The operation invocation blocks until the remote operation completes and the results are returned.

**One-way (`invokeOneWay` method)** The operation invocation does not block while the remote operation is executing, no results are returned. See section 2.3.3 for additional requirements when the underlying protocol is SOAP over HTTP.

**R 2.5 (Failed `invoke`)** When an operation is invoked using the `invoke` method, an implementation MUST throw a `java.rmi.RemoteException` if an error occurs during the remote invocation.

**R 2.6 (Failed `invokeOneWay`)** When an operation is invoked using the `invokeOneWay` method, an implementation MUST throw a `JAXRPCException` if an error occurs during the remote invocation.

Once an operation has been invoked the values of the operation's output parameters may be obtained using the following methods:

**`getOutputParams`** The returned `Map` contains the output parameter values keyed by name. The type of the key is `String` and the type of the value depends on the mapping between XML and Java types.

**`getOutputValues`** The returned `List` contains the values of the output parameters in the order specified for the operation. The type of the value depends on the mapping between XML and Java types.

**R 2.7 (Missing invocation)** An implementation MUST throw `JAXRPCException` if `getOutputParams` or `getOutputValues` is called prior to `invoke` or following `invokeOneWay`.

### 2.3.3 SOAP/HTTP Requirements

The following additional requirements apply when the underlying protocol is SOAP over HTTP:

**R 2.8 (`invokeOneWay` and SOAP/HTTP)** When executing an `invokeOneWay`, an implementation MUST block until the HTTP response is received or an error occurs. Completion of the HTTP request simply means that the transmission of the request is complete, not that the request was accepted or processed.

### 2.3.4 Example

## 2.4 Standard Properties

Table 2.2 lists a set of standard properties that may be set on a `Call` instance and shows which properties are optional for implementations to support.

**R 2.9 (Required `Call` properties)** An implementation MUST support the properties shown as mandatory in table 2.2.

**R 2.10 (Optional `Call` properties)** An implementation MAY support the properties shown as optional in table 2.2.

**R 2.11 (Additional `Call` properties)** An implementation MAY support additional implementation specific properties not listed in table 2.2. Such properties MUST NOT use the `javax.xml.rpc` prefix in their names.

| Name | Type | Mandatory | Description |
|---|---|---|---|
| **`javax.xml.rpc.security`** | | | |
| `.username` | String | Y | Username for HTTP basic authentication. Section **??** describes the JAX-RPC support for HTTP authentication. |
| `.password` | String | Y | Password for HTTP basic authentication. |
| **`javax.xml.rpc.session`** | | | |
| `.maintain` | Boolean | Y | Used by a client to indicate whether it is prepared to participate in a service endpoint initiated session. The default value is `false`. Section **??** describes the JAX-RPC support for HTTP sessions. |
| **`javax.xml.rpc.soap.operation` Properties** | | | |
| `.style` | String | N | The style of the operation: either `rpc` or `document`. An implementation may choose to disallow setting this property and if so will throw `JAXRPCException` if this is attempted. |
| **`javax.xml.rpc.soap.http.soapaction`** | | | |
| `.use` | Boolean | N | |
| `.uri` | String | N | |
| **`javax.xml.rpc.encodingstyle.namespace`** | | | |
| `.uri` | String | N | |

Table 2.2: Standard `Call` properties.

# Appendix A

# Conformance Requirements

# Bibliography


[1] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler. Extensible Markup Language (XML) 1.0 (Second Edition). Recommendation, W3C, October 2000. See http://www.w3.org/TR/2000/REC-xml-20001006.

[2] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Nielsen, Satish Thatte, and Dave Winer. Simple Object Access Protocol (SOAP) 1.1. Note, W3C, May 2000. See http://www.w3.org/TR/SOAP/.

[3] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen. SOAP Version 1.2 Part 1: Messaging Framework. Recommendation, W3C, June 2003. See http://www.w3.org/TR/2003/REC-soap12-part1-20030624.

[4] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen. SOAP Version 1.2 Part 2: Adjuncts. Recommendation, W3C, June 2003. See http://www.w3.org/TR/2003/REC-soap12-part2-20030624.

[5] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1. Note, W3C, March 2001. See http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[6] Rahul Sharma. The Java API for XML Based RPC (JAX-RPC) 1.0. JSR, JCP, June 2002. See http://jcp.org/en/jsr/detail?id=101.

[7] Roberto Chinnici. The Java API for XML Based RPC (JAX-RPC) 1.1. Maintenance JSR, JCP, August 2003. See http://jcp.org/en/jsr/detail?id=101.

[8] Keith Ballinger, David Ehnebuske, Martin Gudgin, Mark Nottingham, and Prasad Yendluri. Basic Profile Version 1.0. Board Approval Draft, WS-I, July 2003. See http://www.ws-i.org/Profiles/Basic/2003-06/BasicProfile-1.0-BdAD.html.

[9] Joseph Fialli and Sekhar Vajjhala. The Java Architecture for XML Binding (JAXB). JSR, JCP, January 2003. See http://jcp.org/en/jsr/detail?id=31.

[10] Joseph Fialli and Sekhar Vajjhala. The Java Architecture for XML Binding (JAXB) 2.0. JSR, JCP, August 2003. See http://jcp.org/en/jsr/detail?id=222.

[11] Roberto Chinnici, Martin Gudgin, Jean-Jacques Moreau, and Sanjiva Weerawarana. Web Services Description Language (WSDL) Version 1.2 Part 1: Core Language. Working Draft, W3C, June 2003. See http://www.w3.org/TR/2003/WD-wsdl12-20030611.

[12] Joshua Bloch. A Metadata Facility for the Java Programming Language. JSR, JCP, August 2003. See http://jcp.org/en/jsr/detail?id=175.

[13] Jim Trezzo. Web Services Metadata for the Java Platform. JSR, JCP, August 2003. See http://jcp.org/en/jsr/detail?id=181.

[14] Jim Knutson and Heather Kreger. Web Services for J2EE. JSR, JCP, September 2002. See http://jcp.org/en/jsr/detail?id=109.

[15] Nataraj Nagaratnam. Web Services Message Security APIs. JSR, JCP, April 2002. See http://jcp.org/en/jsr/detail?id=181.

[16] Farrukh Najmi. Java API for XML Registries 1.0 (JAXR). JSR, JCP, June 2002. See http://www.jcp.org/en/jsr/detail?id=93.

[17] Keith Ballinger, David Ehnebuske, Chris Ferris, Martin Gudgin, Marc Hadley, Anish Karmarkar, Canyang Kevin Liu, Mark Nottingham, Jorgen Thelin, and Prasad Yendluri. Basic Profile Version 1.1. Working Group Draft, WS-I, July 2003. Not yet published.

[18] Martin Gudgin, Amy Lewis, and Jeffrey Schlimmer. Web Services Description Language (WSDL) Version 1.2 Part 2: Message Patterns. Working Draft, W3C, June 2003. See http://www.w3.org/TR/2003/WD-wsdl12-patterns-20030611.

[19] Jean-Jacques Moreau and Jeffrey Schlimmer. Web Services Description Language (WSDL) Version 1.2 Part 3: Bindings. Working Draft, W3C, June 2003. See http://www.w3.org/TR/2003/WD-wsdl12-bindings-20030611.

[20] T. Berners-Lee, R. Fielding, and L. Masinter. RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax. RFC, IETF, March 1997. See http://www.ietf.org/rfc/rfc2396.txt.

[21] S. Bradner. RFC 2119: Keywords for use in RFCs to Indicate Requirement Levels. RFC, IETF, March 1997. See http://www.ietf.org/rfc/rfc2119.txt.

[22] John Cowan and Richard Tobin. XML Information Set. Recommendation, W3C, October 2001. See http://www.w3.org/TR/2001/REC-xml-infoset-20011024/.

[23] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. XML Schema Part 1: Structures. Recommendation, W3C, May 2001. See http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/.

[24] Paul V. Biron and Ashok Malhotra. XML Schema Part 2: Datatypes. Recommendation, W3C, May 2001. See http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.