

# JSR 172 Web Services-XML

Version 0.4, Draft



JSR 172

## COPYRIGHT

### **Samsung Electronics Co. Ltd.**

This material is copyrighted by Samsung Electronics. Any unauthorized reproductions, use or disclosure of this material, or any part thereof, is strictly prohibited and is a violation under the Copyright Law. Samsung Electronics reserves the right to make changes in specifications at any time and without notice. The information furnished by Samsung Electronics in this material is believed to be accurate and reliable, but is not warranted true in all cases.

### **Trademarks and Service Marks**

The Samsung Logo is the trademark of Samsung Electronics. Java is the trademark of Sun Microsystems.

All other company and product names may be trademarks of the respective companies with which they are associated.

## About This Document

This document describes JSR 172 web services with some sample code snippets of XML parsing.

### Scope

This document is intended for Java ME developers who wish to develop Java ME applications. It assumes good knowledge of java programming language.

To know about Java ME basics and Java programming language, refer to the Knowledge Base under Samsung Mobile Innovator (SMI).

<http://innovator.samsungmobile.com/platform.main.do?platformId=3>

## Table of Contents

|   |    |
|---|----|
| Introduction.....   | 5  |
| Overview.....   | 5  |
| Core Specification in Java ME .....                           | 5  |
| Web Services Architecture into Java ME .....                  | 6  |
| JSR 172 Web services consumer: .....                          | 7  |
| The network: .....  | 7  |
| Back-end Web services:.....                                   | 7  |
| Invocation model and data flow as standard Web services ..... | 7  |
| Java ME Web Services Specification – RPC.....                 | 8  |
| Java ME Web Services Specification – XML .....                | 8  |
| JSR 172 - RPC API Description .....                           | 9  |
| JSR 172 - XML Parsing API Description.....                    | 10 |
| XML Parsing Code Snippets: .....                              | 11 |
| DefaultHandler.....   | 11 |
| Get Start and End event notification of XML file .....        | 12 |
| Get Start and End of Tags of XML files.....                   | 12 |
| Parse XML file.....   | 13 |
| Sample code snippet: .....                                    | 13 |

## Table of Figures

|  |   |
|--|---|
| Figure 1: Java ME Web Service Architecture ..... | 6 |
| Figure 2: JSR 172 invocation model.....          | 7 |

## Introduction

This article gives introduction of XML & RPC web services of Java ME, both are optional packages. These two packages are the strip down version of JAX-RPC 1.1 (Java API for XML-Based RPC) and Simple API for XML, version 2 (SAX2) those are targeted for J2SE & J2EE. The Java ME based WSA API provide two independent capabilities into Java ME platform.

1. Remote service innovations
2. XML Parsing

## Overview

The Java ME Web Services API is strip down version of Java SE (J2SE) API of JAX-RPC 1.1 (Java API for XML-Based RPC) and Simple API for XML, version 2 (SAX2). It includes limited Remote Method Innovation (RMI) API to satisfy the JAX-RPC dependency. JSR 172 WSA includes the XML – parsing which is Simple API for XML, version 2 (SAX2).

The goal of WSA is to integrate fundamental support for web services invocation and XML parsing into the device's runtime environment, so application don't need to include the external API for such functionality -- it often expansive into the resource constraint device like mobile phones.

## Core Specification in Java ME

The core specifications and application-level protocols that define web services are promoted by the Web Services Interoperability Organization (WS-I), and governed by the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS). Four key standards specify how to create, deploy, find, and use web services:

- Simple Object Access Protocol (SOAP) 1.1,  
defines transport and data encoding.
- Web Services Definition Language (WSDL) 1.1,

defines how to describe remote services.

- XML 1.0, XML Schema,

defines the XML markup language; and XML schema.

- Universal Description, Discovery, & Integration (UDDI) 2.0

defines how remote services are discovered

The first version of WSA addresses only the consumption of web services. It does not support creation and deployment of service endpoints; a Java ME device can be a service consumer, but not a service producer. JSR 172 also does not support the Universal Description, Discovery, and Integration (UDDI) 2.0 specification that define how remote services are discovered. Because there are quite a few specifications that cover different technologies related to Web services, and the number keeps growing, the Web Services Interoperability Organization (WS-I) has defined the WS-I Basic Profile, Version 1.0 to define the minimum set of Web services specifications, as well as conformance rules that all basic Web services providers and consumers must follow. The JSR 172 specification conforms to the WS-I Basic Profile.

## Web Services Architecture into Java ME

WSA Specifies standardized client-side technology to enable Java ME applications to consume remote services on typical web services architectures as show figure below

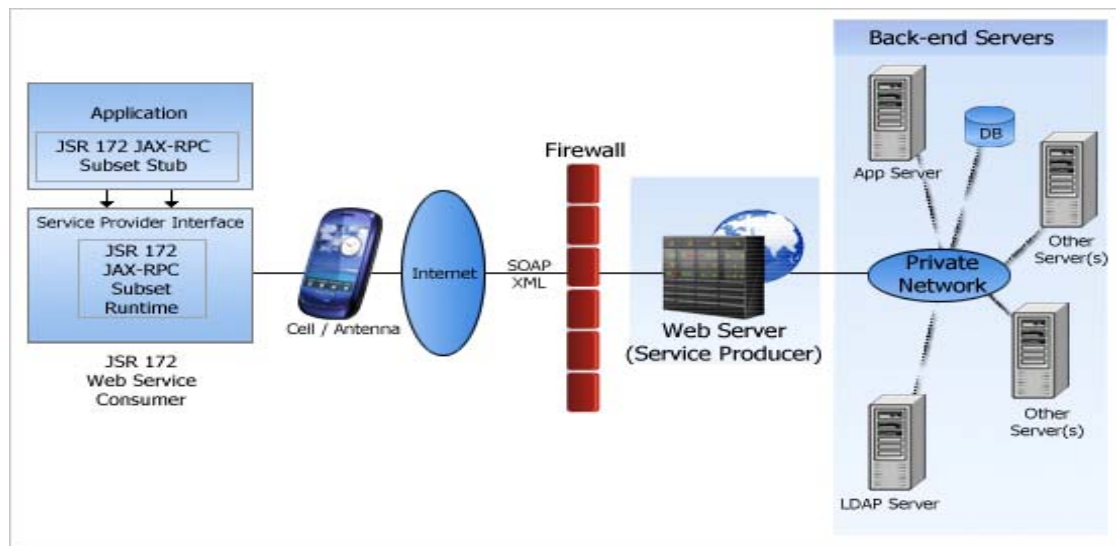


Figure 1: Java ME Web Service Architecture

## JSR 172 Web services consumer:

This is the Java ME application, such as MIDP or Personal Profile-based application, a JSR 172 stub and supporting classes, and the JSR 172 runtime.

### The network:

The wireless network and the Internet, and the corresponding communication and data-encoding protocols, including binary protocols, HTTP, and SOAP/XML.

### Back-end Web services:

A web server, acting as the service producer, typically behind one or more firewalls and a proxy gateway. The web server often provides access to back-end applications and servers on a private network.

## Invocation model and data flow as standard Web services

Java ME applications invoke remote services through the JSR 172 stub(s) and runtime, and typically over HTTP and SOAP. The stubs and runtime hide the complexities associated with remote service invocation including how input and return values are encoded and decoded, and the complexities related to managing network communication to the server. Method invocation follows the synchronous request-response model, as explain below:

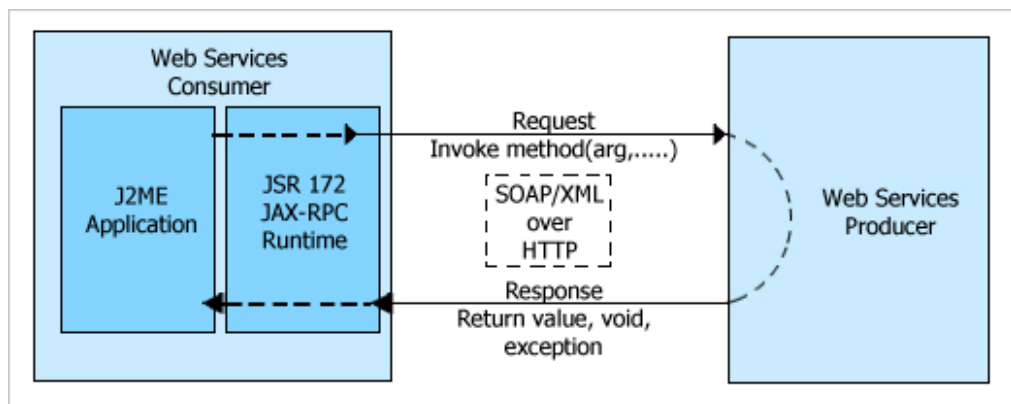


Figure 2: JSR 172 invocation model

## [Java ME Web Services Specification – RPC](#)

JSR 172 RPC is web services remote invocation API is based on J2SE's JAX-RPC 1.1 API. The mention below list describe at high level:

- Conforms to the WS-I Basic Profile
- Supports SOAP 1.1
- Supports any transport, such as HTTP 1.1, that can deliver SOAP messages, and that has a defined protocol binding for SOAP 1.1
- Supports a full suite of data types: boolean, byte, short, int, long, float, double, String, arrays of primitive types, and complex types (structures containing primitive or complex types)
- Supports the Literal representation of a SOAP message representing an RPC call or response (a WSDL operation using the Document/Literal messaging mode); doesn't support Encoded representation
- Doesn't support SOAP Messages with Attachments.
- Doesn't support SOAP message handlers.
- Doesn't support service endpoints; that is, doesn't enable a device to be a web service producer.
- Doesn't service discovery support (UDDI).
- To reduce demands on network bandwidth -- and to save users time and per-byte charges? doesn't mandate the use of XML encoding on the device itself

## [Java ME Web Services Specification – XML](#)

JSR 172 XML is simple subset of Simple API for XML, version 2 (SAX2) ,even though most of the classes in the standard SAX2 API are missing in this subset, all the necessary core functionality remains. The following list describes the JAXP subset at a high level.

- Based on a strict subset of SAX 2.0
- Supports XML namespaces
- Supports both UTF-8 and UTF-16 character encoding
- Doesn't support either Level 1.0 or Level 2.0 of the Document Object Model (DOM), as it's considered to be too heavy.



- Doesn't support Extensible Stylesheet Language Transformations (XSLT)
- Supports Document Type Definition (DTD)

## [JSR 172 - RPC API Description](#)

The table below describes subset of the standard `java.rm.*`; package

| Interface | Description   |
|-----------|---|
| Remote    | The Remote interface serves to identify interfaces whose methods may be invoked from a non-local virtual machine. |

| Exception        | Description  |
|------------------|--|
| MarshalException | A MarshalException is thrown if a <i>java.io.IOException</i> occurs while marshalling the remote call header, arguments or return value for a remote method call.  |
| RemoteException  | A RemoteException is the common superclass for a number of communication-related exceptions that may occur during the execution of a remote method call.   |
| ServerException  | A ServerException is thrown as a result of a remote method invocation when a RemoteException is thrown while processing the invocation on the server, either while unmarshalling the arguments, executing the remote method itself, or marshalling the return value. |

The table below describes subset of the standard `javax.microedition.xml.rpc.*`; package

| Interface          | Description   |
|--------------------|---|
| FaultDetailHandler | The <i>javax.microedition.xml.rpc.FaultDetailHandler</i> interface is implemented by stubs that handle custom faults. |

| class       | Description   |
|-------------|---|
| ComplexType | The class <i>ComplexType</i> is a special Type instance used to represent an xsd: complextype defined in a Web Service's WSDL definition. |

|           |   |
|-----------|---|
| Element   | The class <code>Element</code> is a special Object used to represent an <code>xsd:</code> element defined in a Web Service's WSDL definition.                       |
| Operation | The <i><code>javax.microedition.xml.rpc.Operation</code></i> class corresponds to a <code>wsdl: operation</code> defined for a target service endpoint.             |
| Type      | The class <i><code>Type</code></i> is a type safe enumeration of allowable types that are used to identify simple types defined in a Web Service's WSDL definition. |

| Exception                         | Description  |
|-----------------------------------|--|
| <code>FaultDetailException</code> | The <i><code>FaultDetailException</code></i> class is used to return service specific exception detail values, and an associated <i><code>QName</code></i> , to a Stub instance. |

The table below describes subset of the standard `javax.xml.namespace.*`; package

| class              | Description  |
|--------------------|--|
| <code>QName</code> | <code>QName</code> represents a qualified name as defined in the XML specifications: XML Schema Part2: Datatypes specification, Namespaces in XML, Namespaces in XML Errata. |

## [JSR 172 - XML Parsing API Description](#)

| class                         | Description  |
|-------------------------------|--|
| <code>SAXParser</code>        | Defines the API that represents a simple SAX parser.   |
| <code>SAXParserFactory</code> | Defines a factory API that enables applications to configure and obtain a SAX based parser to parse XML documents. |

| Exception                                 | Description                              |
|---|--|
| <code>ParserConfigurationException</code> | Indicates a serious configuration error. |

|                               |   |
|-------------------------------|---|
| FactoryConfigurationErr<br>or | Thrown when a problem with configuration with the Parser<br>Factories exists. |
|-------------------------------|---|

The table below describes subset of the standard org.xml.sax.\*; package

| Interface  | Description   |
|------------|---|
| Attributes | Interface for a list of XML attributes.                         |
| Locator    | Interface for associating a SAX event with a document location. |

| class       | Description                              |
|-------------|--|
| InputSource | A single input source for an XML entity. |

| Exception                     | Description                                     |
|-------------------------------|---|
| SAXException                  | Encapsulate a general SAX error or warning.     |
| SAXNotRecognizedExcep<br>tion | Exception class for an unrecognized identifier. |
| SAXNotSupportedExcep<br>tion  | Exception class for an unsupported operation.   |
| SAXParseException             | Encapsulate an XML parse error or warning.      |

The table below describes subset of the standard org.xml.sax.helpers.\*; package

| class          | Description                                 |
|----------------|---|
| DefaultHandler | Default base class for SAX2 event handlers. |

## [XML Parsing Code Snippets:](#)

### [DefaultHandler](#)

To get parsed xml data need to extends base class with the DefaultHandler,

```
...
public XMLUtility extends DefaultHandler{
```

```
....
}
...
```

### [Get Start and End event notification of XML file](#)

To get start and end event notification of XML file are get by implementing startDocument () and endDocument () method into the class, these are the callback methods

```
...
public XMLUtility extends DefaultHandler{

public void startDocument(){
    }
    public void endDocument(){
    }
}
...
```

### [Get Start and End of Tags of XML files](#)

To get start and End of tag notification can be get by implementing the startElement(...) and endElement(...) method into the class, these are the callback methods

```
...
public XMLUtility extends DefaultHandler{

public void startElement(String uri, String localName, String qName, Attributes attributes)
throws SAXException{
    }
    public void endElement(String uri, String localName, String qName)throws SAXException{
    }

}
```

```
.....
```

### Parse XML file

To parse XML file need SAXParserFactory instance, it provide the SAXParser object which help to parse XML file.

```
...
SAXParser saxParser = null;

SAXParserFactory saxParserFactory = SAXParserFactory.newInstance();

saxParser = saxParserFactory.newSAXParser();

// A single input source for an XML entity

InputSource inPutSource = new InputSource(inStream);

//It parse the contain given by inputStream as XML using the specified
//DefaultHandler

saxParser.parse(inPutSource, this);//it call the callback methods.....
```

### Sample code snippet:

```
import javax.microedition.io.Connector;
import javax.microedition.io.HttpConnection;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.List;
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.Displayable;
```

```

import java.io.InputStream;
import java.util.Vector;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.InputSource;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.SAXException;
import org.xml.sax.Attributes;

public class XMLParsing extends MIDlet implements CommandListener,Runnable {

    private final Command cmdExit = new Command("Exit",Command.EXIT,1);
    private final Command cmdSelect = new Command("Select",Command.ITEM,1);
    private final Command cmdback = new Command("Back",Command.BACK,1);
    private String rssFeed = "http://www.rediff.com/rss/inrss.xml";
    private List topNews = null;
    private Form news = null;
    private Display display = null;
    private HttpURLConnection httpConn = null;
    private InputStream input = null;
    private XMLUtil xmlUtil = null;
    public static boolean isStart = true;

    public XMLParsing(){
        init();
        xmlUtil = new XMLUtil();
        new Thread(this).start();
    }

    public void startApp() {

    }

    public void pauseApp() {

    }
}

```

```

public void destroyApp(boolean unconditional) {

}

private void init(){

    display = Display.getDisplay(this);
    topNews = new List("RssFeed",List.IMPLICIT);
    topNews.addCommand(cmdExit);
    topNews.addCommand(cmdSelect);
    topNews.setCommandListener(this);
    news = new Form("");
    news.addCommand(cmdback);
    news.setCommandListener(this);
}

public void commandAction(Command cmd, Displayable dis) {
    if(cmd.equals(cmdExit)){
        destroyApp(true);
        notifyDestroyed();
    }else if(cmd.equals(cmdSelect)){
        display.setCurrent(news);
        news.deleteAll();
        news.append(((String)xmlUtil.news.elementAt(topNews.getSelectedIndex() + 1)));
    }else if(cmd.equals(cmdback)){
        display.setCurrent(topNews);
    }
}

public void run() {

    xmlUtil.LoadXML(getInputStream(rssFeed));
    while(isStart){
    }
}

```

```

        if(xmlUtil.newsHeading.size() > 1)
            for(int i=1;i<xmlUtil.newsHeading.size();i++){
                topNews.append((String)xmlUtil.newsHeading.elementAt(i), null);
            }
        display.setCurrent(topNews);
    }
    private InputStream getInputStream(String URL){

        try{
            httpConn = (HttpConnection)Connector.open(URL, Connector.READ_WRITE);
            input = httpConn.openDataInputStream();
        }catch(Exception ex){
            ex.printStackTrace();
        }
        return input;
    }

}

class XMLUtil extends DefaultHandler{

    private String title = "title";
    private String description = "description";
    private boolean isNewsHeading;
    private boolean isNews;
    public Vector newsHeading = null;
    public Vector news = null;

    public XMLUtil(){
        super();
        newsHeading = new Vector();
        news = new Vector();
    }

    public void LoadXML(InputStream inStream){
        SAXParser saxParser = null;

```



```

    try{
        SAXParserFactory saxParserFactory = SAXParserFactory.newInstance();
        saxParser = saxParserFactory.newSAXParser();
        InputSource inPutSource = new InputSource(inStream);
        saxParser.parse(inPutSource, this);

    }catch(Exception ex){

    }
}

public void startElement(String uri, String localName, String qName, Attributes
attributes) throws SAXException {
    if(qName.equals(title))
        isNewsHeading = true;
    else if(qName.equals(description))
        isNews = true;
}

public void endElement(String uri, String localName, String qName)throws
SAXException {

    if(qName.equals(title) )
        isNewsHeading = false;
    else if(qName.equals(description))
        isNews = false;
}

public void startDocument(){

}

public void endDocument(){
    XMLParsing.isStart = false;
}

public void characters(char[] ch, int start, int length) throws SAXException {
    String chars = new String(ch, start, length).trim();
    if(isNewsHeading){

```

```
        newsHeading.addElement(chars);  
    }else if(isNews){  
        news.addElement(chars);  
    }  
}  
}
```