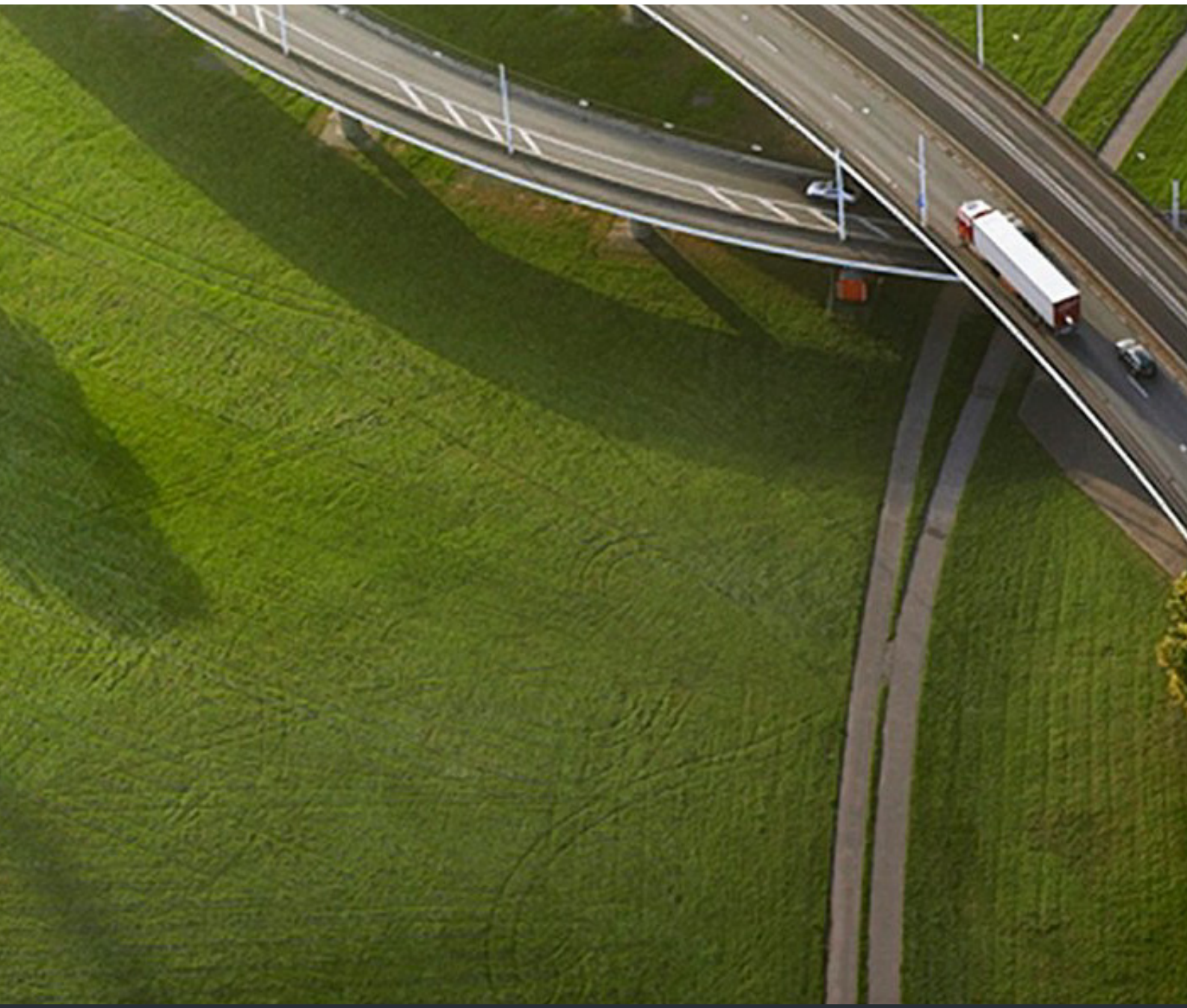
UNIFY Harmonize your enterprise



OpenScape Voice V8
Application Developers Manual

Programming Guide

A31003-H8080-R100-4-7620

Our Quality and Environmental Management Systems are implemented according to the requirements of the ISO9001 and ISO14001 standards and are certified by an external certification company.

Reference No.: A31003-H8080-R100-4-7620

UNIFY Harmonize your enterprise

unify.com

Contents

Contents

1 Important Notices

1.1 About This Book

This manual provides overview and programming information for application development working with the OpenScape Voice V4 and later released product.

1.1.1 Prerequisite Knowledge

Users should have completed the appropriate technical and product training for developing applications working with the OpenScape Voice V4 and later releases.

These manual is intended for use by application developers working with the OpenScape Voice V4 and later release product.

1.1.2 How to Use This Book

Use this manual to complete application development programming tasks desired for the OpenScape Voice V4 and later releases.

1.1.3 Special Notices

If applicable, potentially dangerous situations are noted throughout this guide. The three alert methods are defined below:

DANGER	A danger notice calls attention to conditions that, if not avoided, will result in death or serious injury.
WARNING	A warning notice calls attention to conditions that, if not avoided, could result in death or serious injury.
Caution	A caution notice calls attention to conditions that, if not avoided, may damage or destroy hardware or software.

1.2 Documentation Feedback

When you call or write, be sure to include the following information. This will help identify which document you are having problems with.

- **Title:** OpenScape Voice V8, Application Developers Manual, Programming Guide

- **Order Number:** A31003-H8080-R100-03-7620

1.2.1 For U.S. only

To report a problem with this document, call your next level of support:

- Customers should call the Unify Customer Support Center.
- Unify GmbH & Co KG employees should call the Interactive Customer Engagement Team (i-CET) or use the Document Feedback form that you can access from the front page of the HTML version of this document.

1.2.2 Countries other than U.S.

Please provide feedback on this document as follows:

- Submit a trouble ticket to ICTS, or
- Use the Document Feedback form that you can access from the front page of the HTML version of this document.

2 The Application Developers Manual

The OpenScape Voice V8 Application Developers Manual is subdivided into the following sections.

- Web Services SDK Programming Overview
- Link Failure Management
- Business Group Management
- Subscriber Self-Care

3 About the OpenScape Voice Web Services SDKs

The OpenScape Voice Web Services SDK Programming Overview provides application development programming information that is common to all SDKs, as well as an overview of each SDK.

3.1 Architecture

Bild 1 shows the generic architecture of the Unify Web Services SDKs.

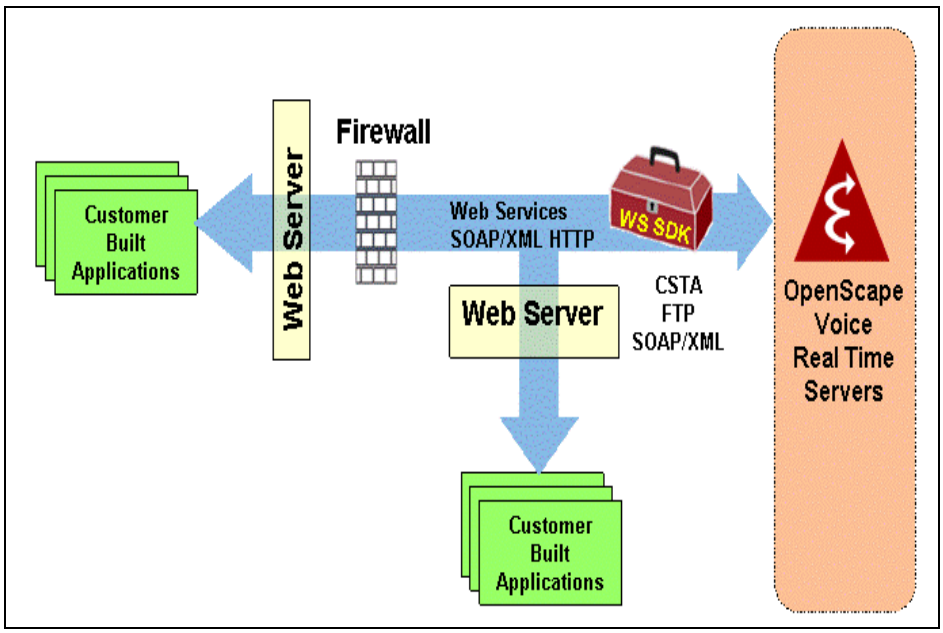


Figure 1 Web Services SDK General Architecture

Each SDK exposes a Web Service Interface. This interface comes from the OpenScape Voice system Real Time Servers.

Customers then build applications that access these Unify provided Web Services. Using the functionality of the Web Services, customers build custom applications to accomplish various tasks.

Because these SDKs are provided as Web Services, they are accessible by Web Servers and custom applications that reside outside the firewall. While this deployment is the most likely scenario, there is no reason the applications and Web Server could not reside within the firewall as well.

There may be numerous customer applications and numerous Web Servers.

For the various SDKs, the Unify SDK Servers host both the Web Service Interface and the Web Service Execution layer.

Attention: For most customers, the SDK Server referenced in this guide will be the OpenScape Voice system. Talk to your Unify representative for details.

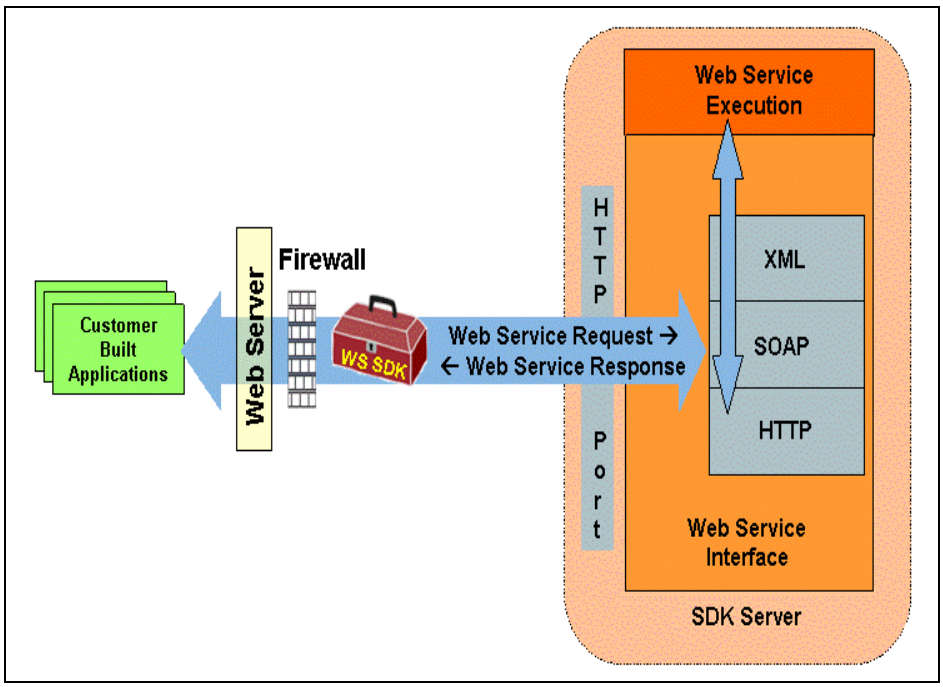


Figure 2 Communication Between Applications and SDK Servers via Web Services

Communications between the applications and the SDK Servers are via Web Services. These web services communicate via SOAP/XML over HTTP using the HTTP port of the SDK Server that is hosting the Web Service Interface implementation.

When an HTTP POST message is received, the SOAP envelope is extracted from the HTTP message. Next the XML data is extracted form the SOAP envelope. At this point the XML message is interpreted by the Web Service Execution layer and the appropriate actions are taken.

The result of executing the request (the response) follows the reverse path back to the application. It is given to the Web Service Interface layer to be packaged into XML, stuffed into a SOAP envelope, and then transported back in an HTTP response message.

Once a customer is ready to develop an application, the SDK WSDL files can be run through any number of industry-standard tools to create proxy files in the desired target language; for example, C++ or Java. Unify recommended environments include the SourceForge gSOAP and Apache Axis environments, but any tool that uses WSDL files can be tried.

Once a set of files has been created in the target language, the programmer uses knowledge obtained from this SDK document and sample applications to create his own custom applications. IMS resources such as forums and newsgroups can be used to get answers to specific questions and/or see answers to common problems.

The final result is a working, custom-built application that uses the specified SDK.

3.2 SDK Package Contents

A Web Services SDK is made up of a set of components and tools that the customer uses to create custom applications. Typically, an SDK will consist of:

- The WSDL file
- An ADG, which includes sample code
- The interface specification, provided in Interface Manual: Volume 3, SOAP/ XML Subscriber Interface Provisioning.

3.3 Applications Development

The target applications development environments for the Link Failure Management SDK are any environments that support C++ or Java applications. Two good tools for generating header and source files from WSDL are SourceForge gSOAP and Apache Axis. It should be noted, however, that environmental and tool differences do arise, and each different environment may require some unique solutions. There are no requirements as to which Web Server is used.

3.3.1 SourceForge gSOAP

gSOAP is an open source SOAP toolkit that facilitates Web Services development in a C/C++ environment. Within this environment is a tool called wsdl2h which will take a WSDL file as input and create the C/C++ header and code files needed to call the Web Services defined by that WSDL file. Detailed information on SourceForge gSOAP can be found at: http://gsoap2.sourceforge.net/.

3.3.2 Apache Axis

Axis is the Apache implementation that supports the SOAP standard as defined by W3C. Within this environment is a tool called wsdl2java which will take a WSDL file as input and create the java code needed to call the Web Services defined by that WSDL file.

Detailed information on Apache Axis can be found at: http://ws.apache.org/axis/.

3.4 Technical Implementation Notes

The following sections describe the technical implementation common to all SDKs.

3.4.1 Discovery of Web Services

There is no explicit discovery mechanism for Web Services SDKs. Web Services SDKs are accessed by using the IP address of the SDK Server that hosts the SOAP server and the HTTP port.

3.4.2 Web Service Request/Response

A Web Service request/response consists of an SOAP/XML-encoded request and response that is transported by HTTP, within the HTTP POST and HTTP 200 OK messages. Each protocol identified as follows: HTTP is in plain text, SOAP is underlined, and *XML is in italics*.

3.4.3 Request-Response Operation

All Unify SDKs currently support only a request-response paradigm. In this paradigm, a request is made and a response specific to that request is returned. The basic flow of the request-response model is as follows, with key components highlighted in **bold**:

1. The **Customer-Built Application** makes a Web Service request.
2. The Web Service request is processed by the **Web Server** and the proper SOAP/XML request is sent through the **Firewall** to the **Web Service Interface** on the correct **Unify SDK Server** via an HTTP POST message.
3. The **Web Service Interface** receives the request, parses the SOAP/XML content, and passes it to the **Web Service Execution** layer for execution.

4. The **Web Service Execution** layer executes the request, formulates the response, and gives it to the **Web Service Interface** layer for encoding and packaging.
5. The **Web Service Interface** layer encodes a SOAP/XML message and sends it back through the **Firewall** to the **Web Server** via an HTTP 200 OK message.
6. The **Web Server** forwards the response to the **Customer-Built Application**, which can then continue its processing.

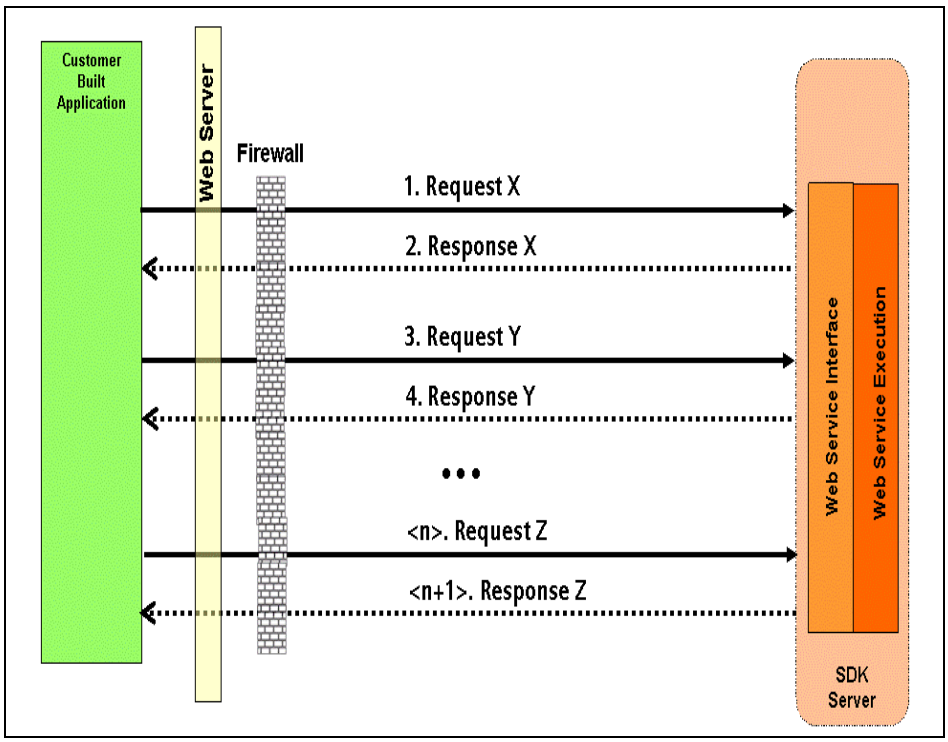


Figure 3 Request-Response Operation

Responses are not correlated to requests with any type of request ID, and as such an application must wait for a response before sending the next request. In other words, on a single thread of the application, no more than one single request can be outstanding at one time.

The Unify Web Services SDKs do not support any kind of monitoring or subscribe-notify paradigm where asynchronous event flow is required.

If a response is not returned within a certain timeframe, the HTTP request will time out and an error will be returned to the application.

3.4.4 Security

The Unify Web Services SDKs do not provide any general mechanisms for security common to all SDKs. Authentication, authorization, and other security mechanisms will be provided on a per-SDK basis as deemed necessary. Refer to the specific SDK documentation for details.

3.4.5 Sessions

With respect to the application context, the Unify Web Services SDKs are sessionless.

3.4.6 Server Restart

There will be no specific indication to the application of a server restart. The only indication an application might see is the timeout of a current request.

3.4.7 Application Restart

The Unify SDK Servers will have no indication of an application restart.

3.4.8 Heartbeat Mechanisms

The Unify SDKs do not support any common heartbeat messages.

3.4.9 Error Handling

Responses may return specific errors relating to a specific request. There is no overall general error message scheme amongst all SDKs. For specific errors, refer to the supporting documentation of each SDK.

Under certain conditions, a Web Service request may time out. This general condition may indicate failure in the network, failure on the server, etc. It does not point to one specific failure.

SOAP and XML processing errors may also be returned. They indicate badly formed SOAP or XML messages, which typically indicate a customer application problem.

4 Description of Web Services SDKs

This chapter describes the set of SDKs that are currently provided and supported by Unify for the OpenScape Voice environment.

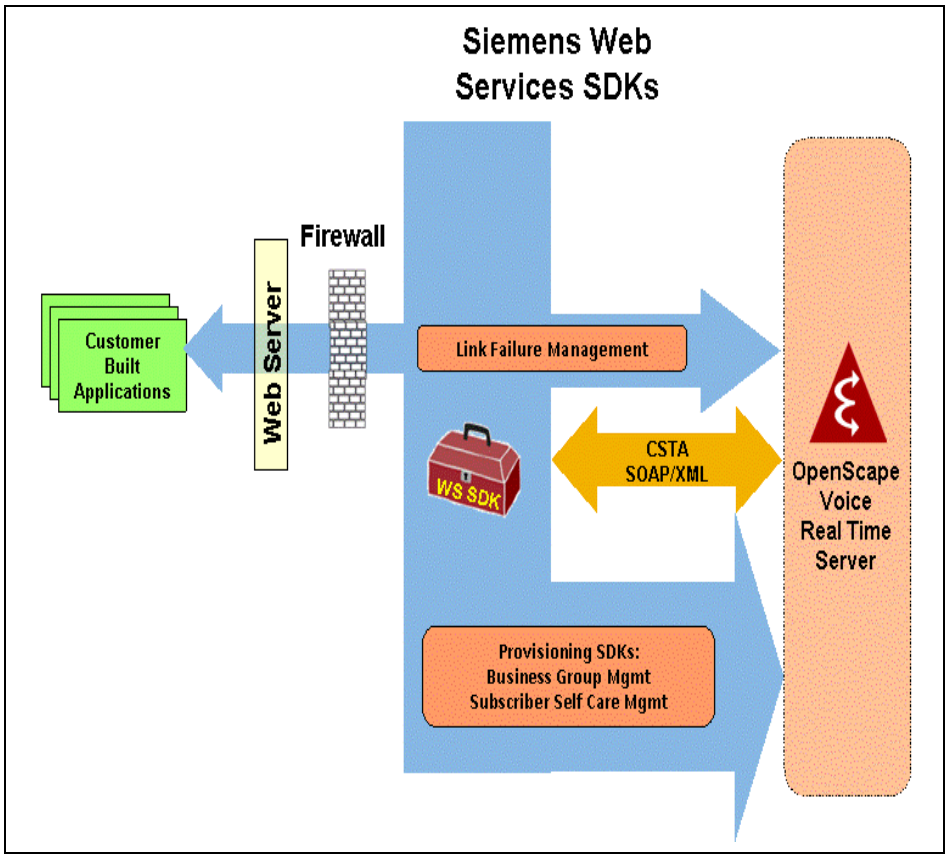


Figure 4 Current Set of Unify WS SDKs for OpenScape Voice

There are several SDKs provided by the OpenScape Voice system Real Time Server. Bild 4 shows which SDKs are exposed by which components.

Taken together, these SDKs make up the Web Services SDKs supported by Unify for the OpenScape Voice system.

4.1 Link Failure Management SDK on the OpenScape Voice

The following SDK is provided by the OpenScape Voice system real time server to facilitate the management of link failures.

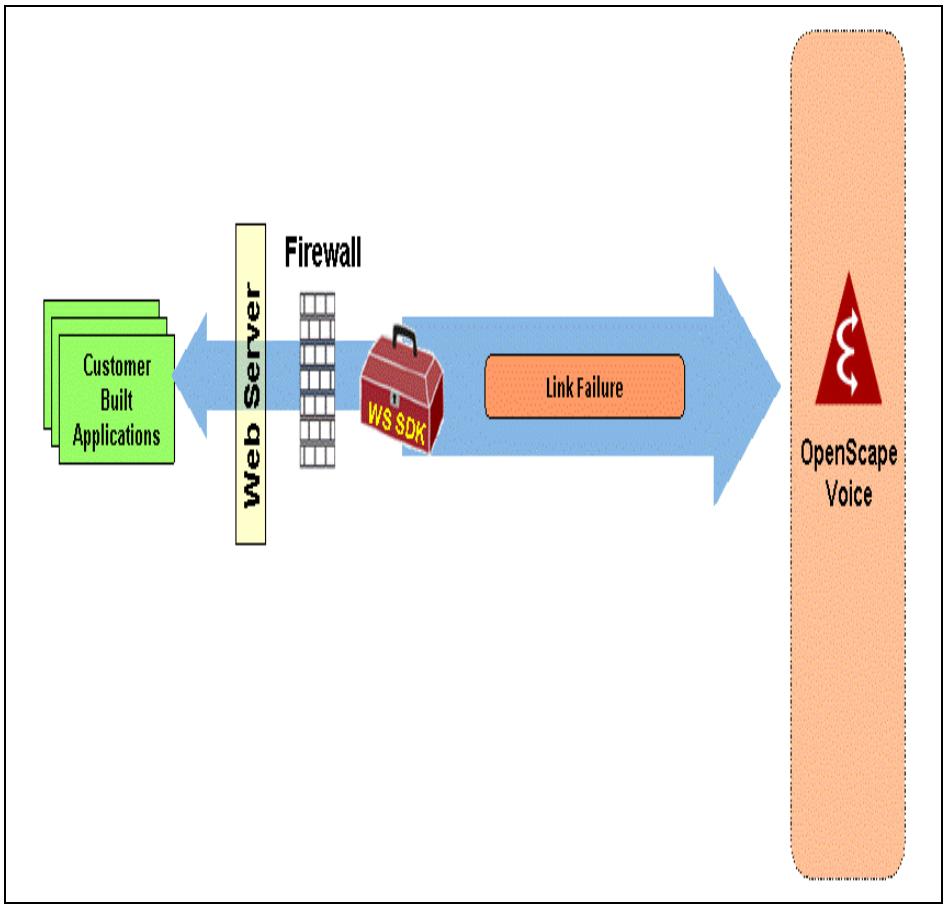


Figure 5 OpenScape Voice Link Failure SDK

With the Web Service for Link Failure Management, a Network Management Service (NMS) application can inform the OpenScape Voice system that an access link is down and instead a backup link will be used, or vice versa. With this information, an NMS application can always use the appropriate link capacity for its call admission calculations.

4.2 SDKs for Provisioning

The following set of SDKs will be provided by the OpenScape Voice system Real Time Server to facilitate Subscriber and Business Group management.

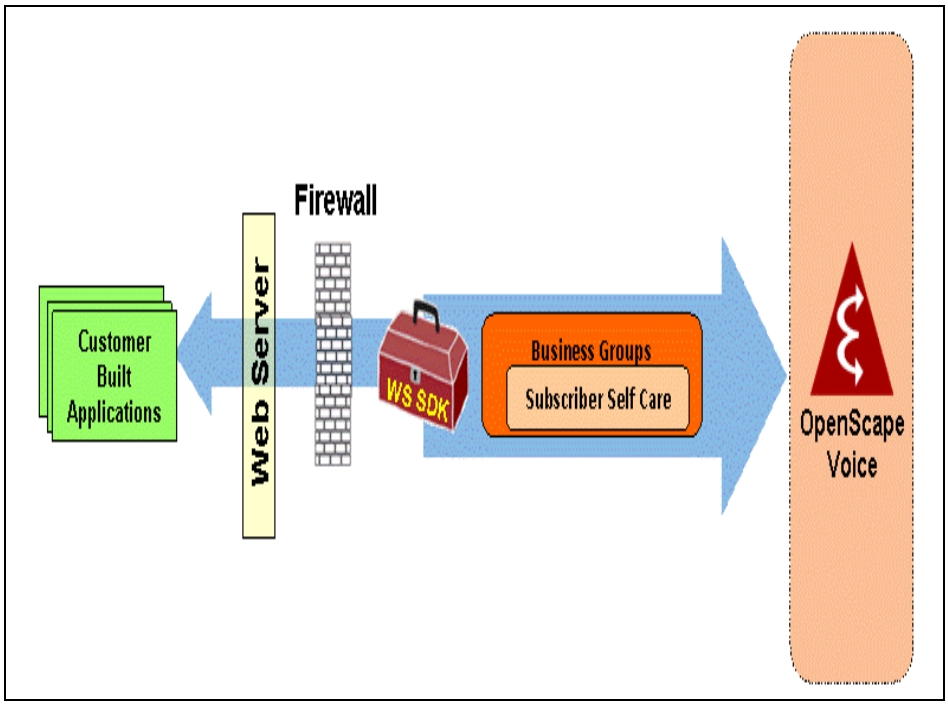


Figure 6 OpenScape Voice Provisioning SDKs

This is a set of layered SDKs. The outside layer is the Business Groups SDK, which adds its own functionality and contains the Subscriber Self Care SDK.

This set of SDKs supports the following functionality:

- ***Business Group Management***
  This SDK provides functions associated with managing business groups, BGLs, and numbering plans.
- ***Subscriber Self Care***
  This SDK provides functions associated with subscriber self care (SSC).

5 Generating a Java client Proxy and a Sample Application

The following example demonstrates how to create a Java client proxy and a sample application using the Axis2 runtime environment based on Unify's WSDL file (`OpenScape_Voice.wsdl`).

Prerequisites:

- **Eclipse IDE for Java EE Developers**
  `https://www.eclipse.org/downloads/`

  Note: `eclipse-jee-kepler-SR2-win32-x86_64.zip` was used for this guide.

- **Apache Axis2 v1.6.2**
  `http://ws.apache.org/axis2/download.cgi`
- **Axis2 Eclipse Codegen Plugin**
  `http://axis.apache.org/axis2/java/core/tools/eclipse/wsdl2java-plugin.html`

  Download and install **Code Generator Wizard for Eclipse Plug-in**,

  Note: v1.6.2 was used for this guide.

  and follow the instructions that make use of the **dropins** directory.
  `http://axis.apache.org/axis2/java/core/tools/eclipse/plugin-installation.html`
- Download **Apache Tomcat**
  `https://tomcat.apache.org/`

  Note: Tomcat 6.0 was used for this guide.

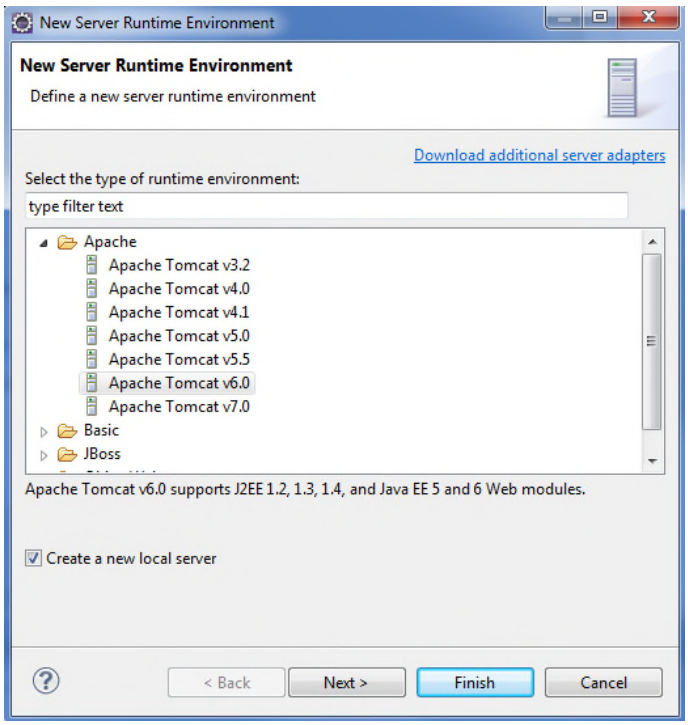
  and extract it on a local folder.

5.1 Eclipse IDE configuration

Configure **Eclipse** to support **Apache Tomcat Server** and **Apache Axis2**, and verify that **Axis2 Eclipse Codegen Plugin** has been installed successfully:

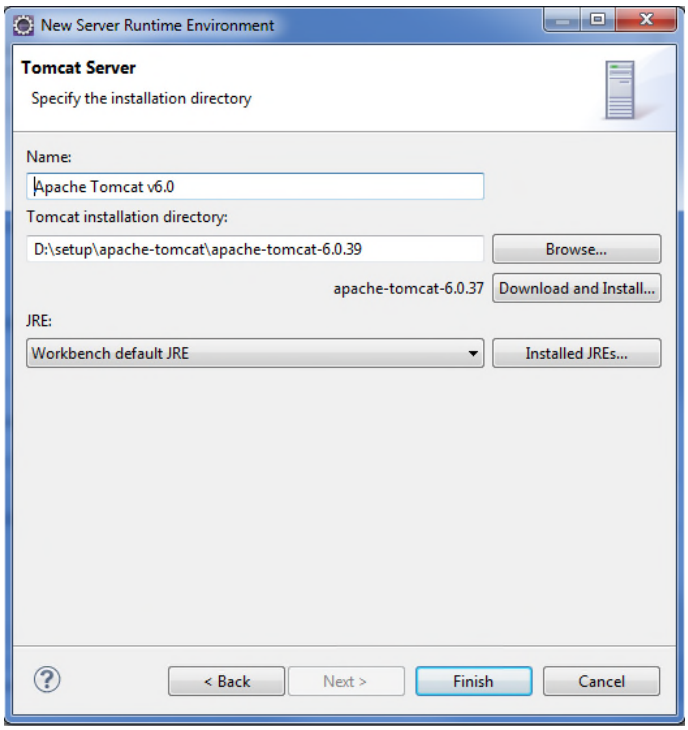
1. Start the **Eclipse WTP** workbench.
2. Navigate to **Window** > **Preferences** > **Server** > **Runtime Environment** and click **Add...**

3. In the **New Server Runtime Environment** window select your runtime environment under the **Apache** category and mark the **Create new local server** checkbox.

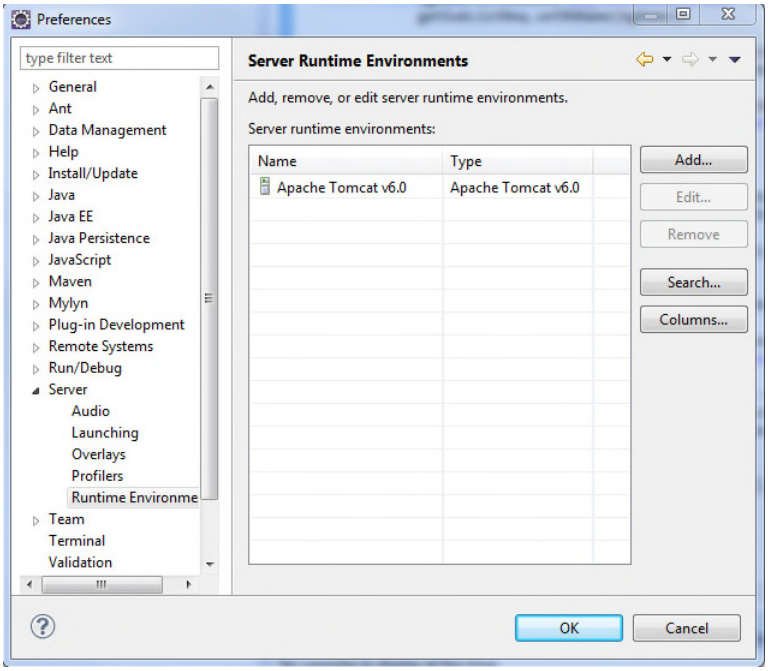


Click **Next >**

4. Fill in your Tomcat installation directory on the next screen and click **Finish**.

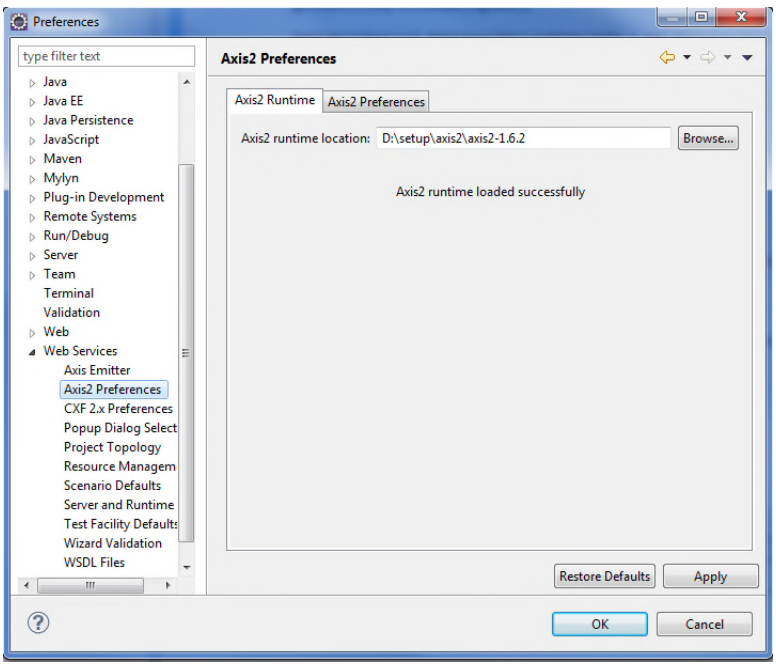


5. `Apache Tomcat v6.0` is now listed under Server Runtime Environments:



Click **OK**.

6. Navigate to **Window** > **Preferences** > **Web Services** > **Axis2 Preferences**. Select the **Axis2 Runtime** tab and point to the correct Axis2 runtime location.



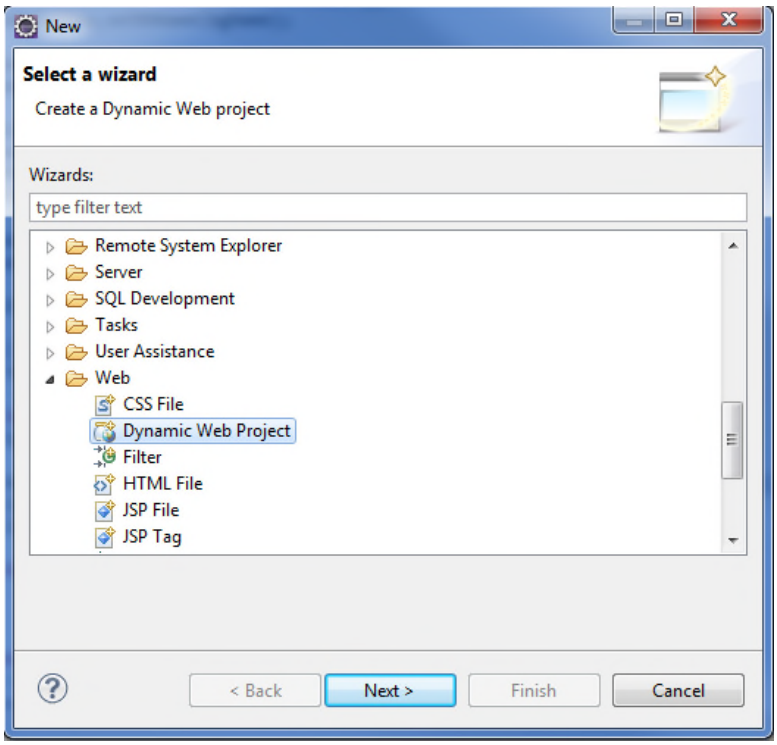
Click **OK**.

7. To verify that **Axis2 Eclipse Codegen Plugin** has been successfully installed go to **File** > **New** > **Other…** and make sure that under the **Axis2 Wizards** category, `Axis2 Code Generator` is listed.

5.2 Java client proxy creation

To create a Java client proxy from a WSDL file (`OpenScape-Voice.wsdl`) do the following:

1. Navigate to **File** > **New** > **Other...**
2. Under the **Web** category, select `Dynamic Web Project` and click **Next >**.



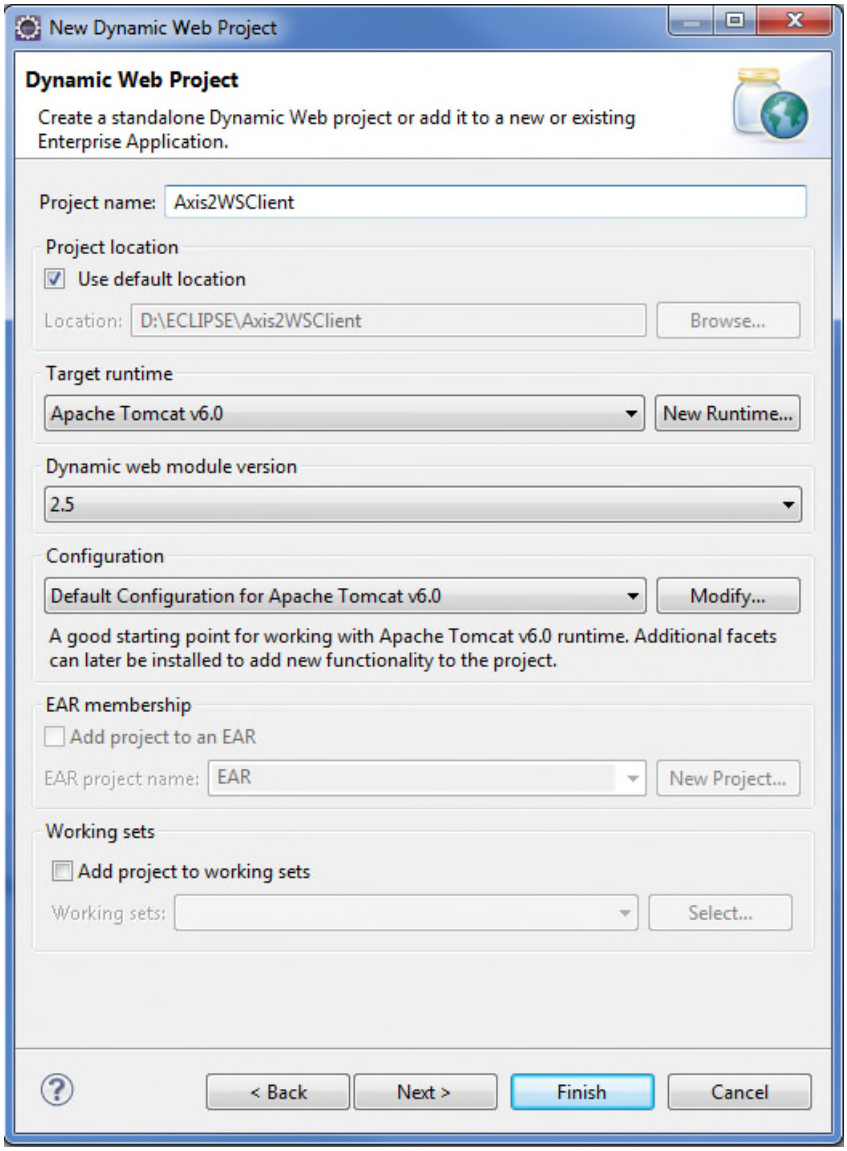
3. In the **New Dynamic Web Project** screen:

   a) Type in the **Project name** (i.e. `Axis2WSClient`)

   b) Make sure that `Apache Tomcat v6.0` is the selected **Target runtime**.

   c) Click on **Modify...**

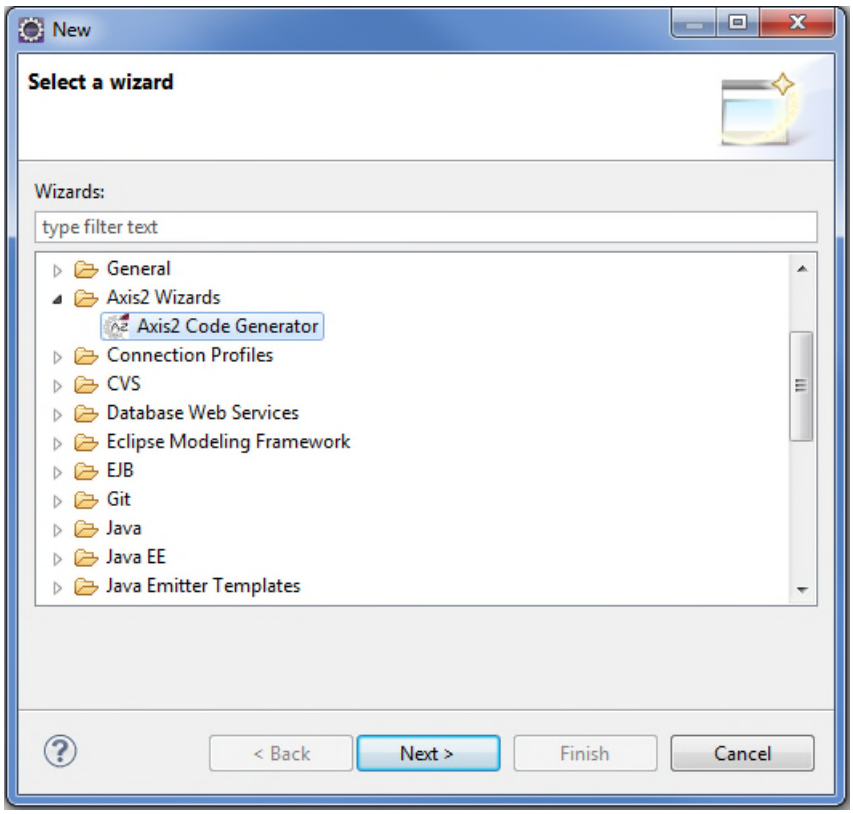
   d) Select the `Axis2 Web Services` project facet and click **OK**.



   e) Click on **Finish**.

   A dynamic web project is created in the workbench.

4. To create a Java client stub, the **Axis2 Eclipse Codegen Plugin** (`wsdl2java`) will be used. Go to **File** > **New** > **Other…** > **Axis2 Wizards** > **Axis2 Code Generator**.



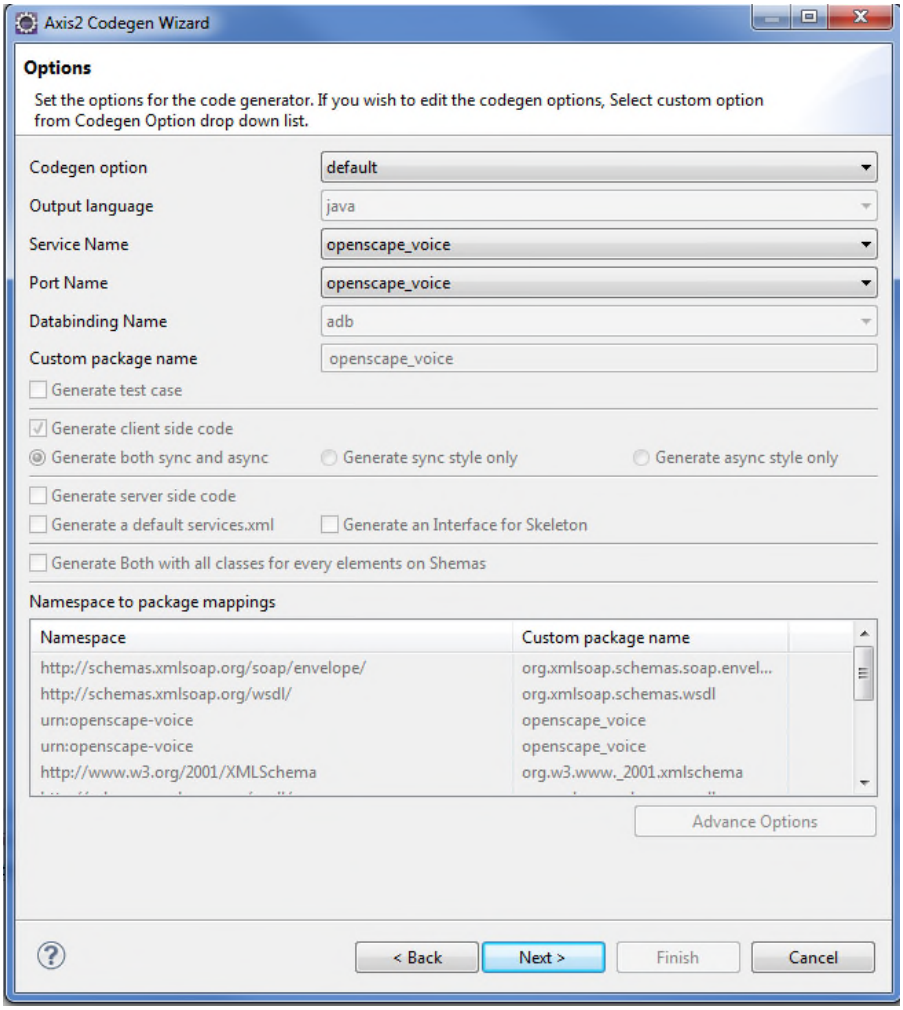
Click **Next >**.

5. On the next screen select **Generate Java source code from a WSDL file** and click **Next >**.

6. Click on **Browse...** and point to the location where Unify's WSDL file (`OpenScape-Voice.wsdl`) is stored.

   Click **Next >**.

7. On the next screen use the default values



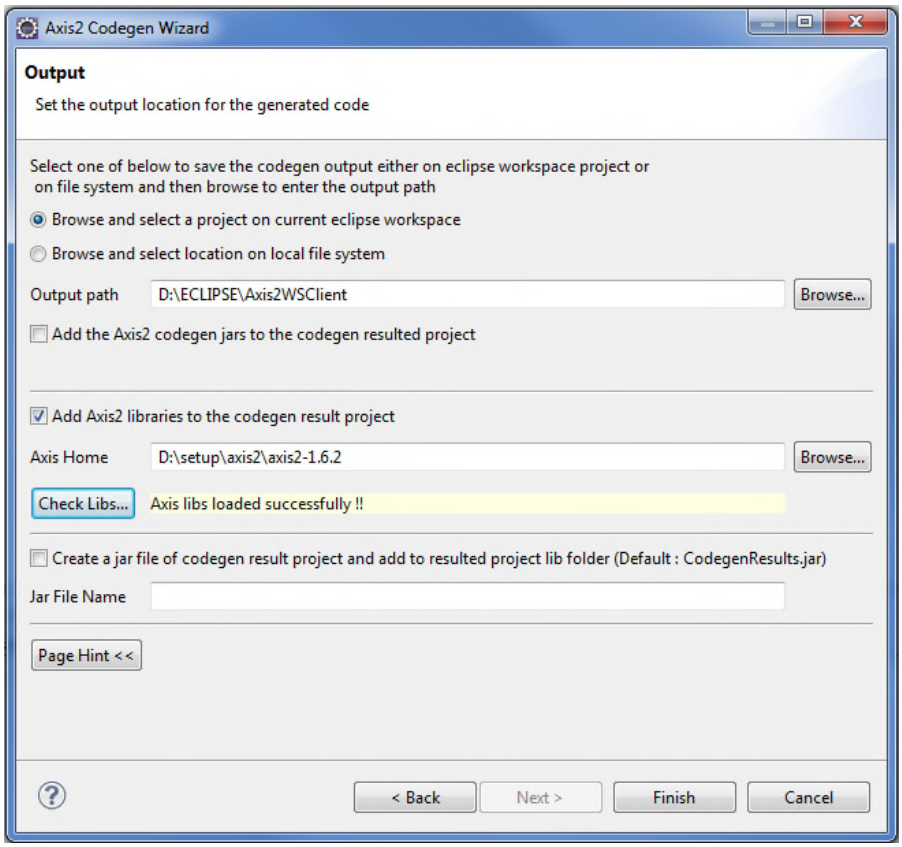
and click **Next >**.

8. In the **Axis2 Codegen Wizard** screen:

   a) Select **Browse and select a project on current eclipse workspace** and click on **Browse...** Then point to the Axis2WSClient directory.

   b) Mark the **Add Axis2 libraries to the codegen result project** checkbox and click on the **Browse...** button.

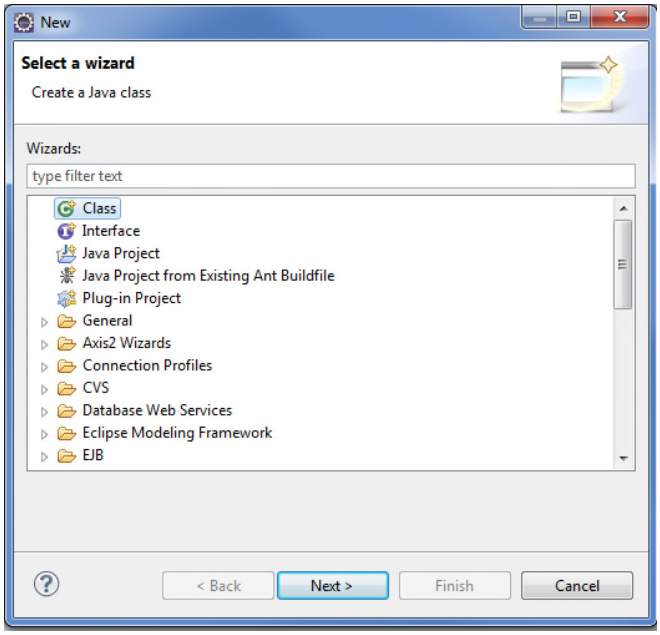
   c) Point to the directory where the downloaded Axis2 libraries are stored and then click on the **Check Libs...** button to verify that the operation completed successfully.



9. All client stub and library files are now generated and stored under the …/Axis2WSClient\src\openscape_voice and …/Axis2WSClient\lib directories respectively.

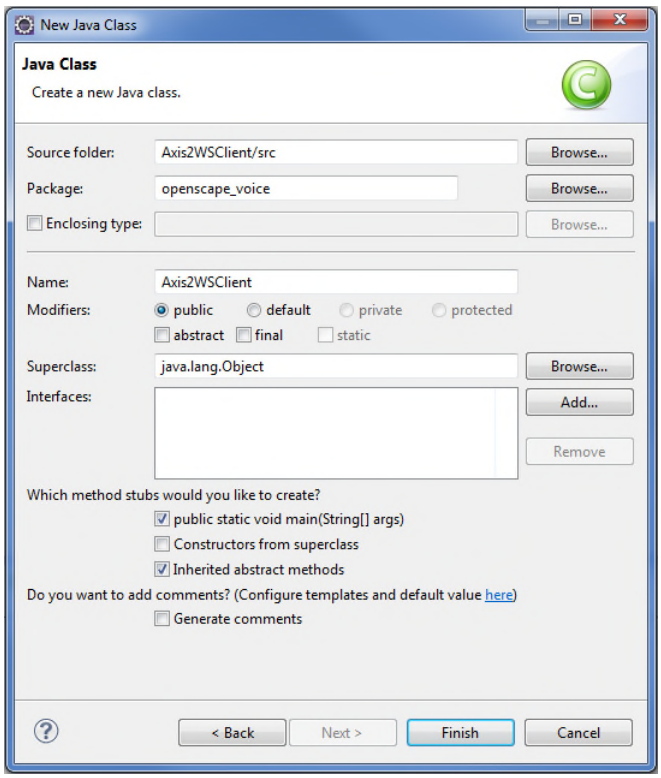
10. Press **F5** to refresh the project.

11. Navigate to **File** > **New** > **Other...** > **Class** to create your main function.



Click **Next >**.

12. Set the name of the class to `Axis2WSClient` and mark the **public static_void main(Strings[] args)** checkbox.



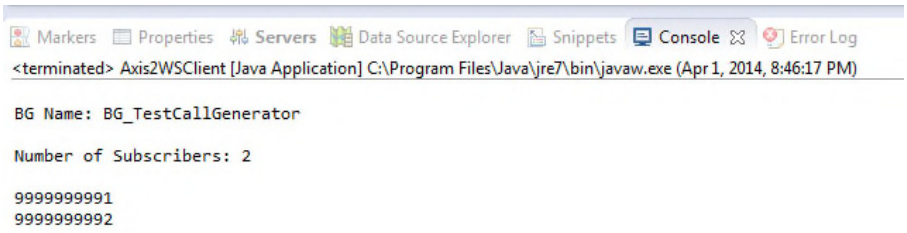
Click **Finish**.

13. You are now ready to use the main java stub (`Openscape_voiceStub.java`) to create your client. The example below demonstrates how to retrieve the number of subscribers belonging to a specific Business Group (`BG_TestCallGenerator`).

```
package openscape_voice;

import java.rmi.RemoteException;
import org.apache.axis2.AxisFault;

public class Axis2WSClient{

    {public static void main(String[] args) {
        try {
            Openscape_voiceStub osv = new Openscape_voiceStub("http://10.7.182.10:8767");

            GetSubscriberListRequest getSubListReq = new GetSubscriberListRequest();

            BGName bgName = new BGName();
            bgName.setBGName("BG_TestCallGenerator");
            getSubListReq.setBGName(bgName);

            GetSubscriberList GetSubscriberListEntryInfo = new GetSubscriberList();
            GetSubscriberListEntryInfo.setGetSubListReq(getSubListReq);

            HiqGLOBALHEADER globalHeader = new HiqGLOBALHEADER();
            globalHeader.setOperatorId("Axis2WSClient");
            HiqHEADER hiqHEADER = new HiqHEADER();
            hiqHEADER.setHiqHEADER(globalHeader);

            GetSubscriberListResult result = new GetSubscriberListResult();
            GetSubscriberListData listData = new GetSubscriberListData();

            result.setGetSubscriberListData(listData);

            try {
                result = osv.getSubscriberList(GetSubscriberListEntryInfo, hiqHEADER);

                int noOfSubs = result.getTotalSubsFound();
                System.out.println("\nBG Name: " + bgName);
                System.out.println("\nNumber of Subscribers: " + noOfSubs + "\n");

                GetSubscriberListEntry[] listEntries = new GetSubscriberListEntry[noOfSubs];
                listData = result.getGetSubscriberListData();
                listEntries = listData.getGetSubscriberListEntry();

                for(int i=0; i<noOfSubs; i++)
                    System.out.println(listEntries[i].getServiceId());

            } catch (RemoteException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        } catch (AxisFault e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

14. Select **Run** > **Run As** > **Java Application**.

    Check the output on the console.



6 Generating a Service Reference and a Sample Application

The following example demonstrates how to consume a **WebService** using a **WSDL** file in **C#**.

Prerequisites:

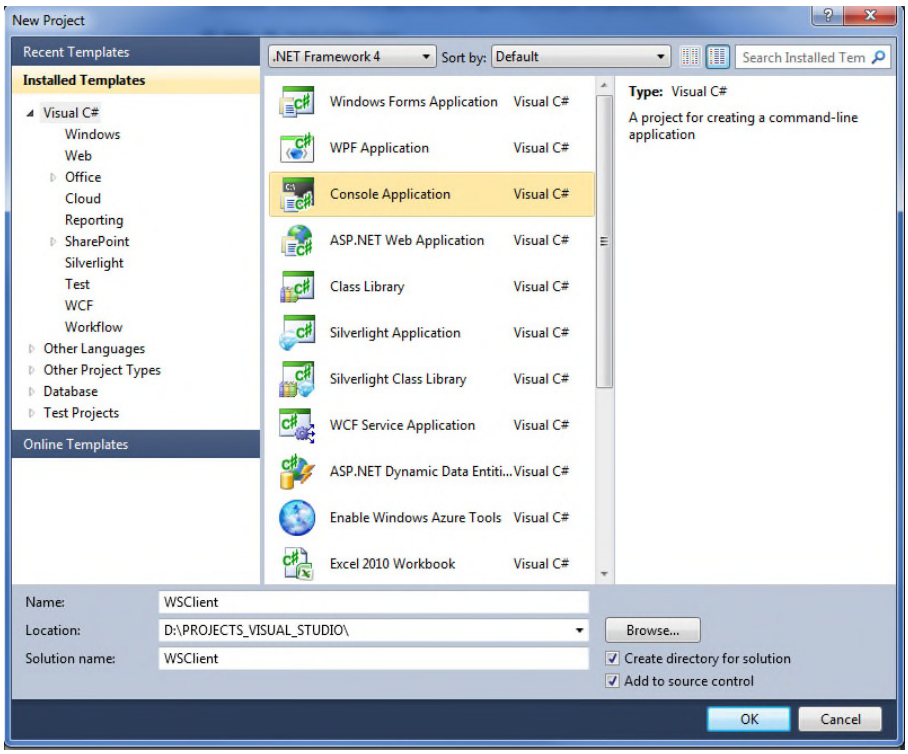
- Visual Studio 2010 Professional installed

6.1 Creating a Visual C# project

To create a Visual C# project to consume the provided WSDL file (`OpenScape Voice.wsdl`), follow the procedure below:

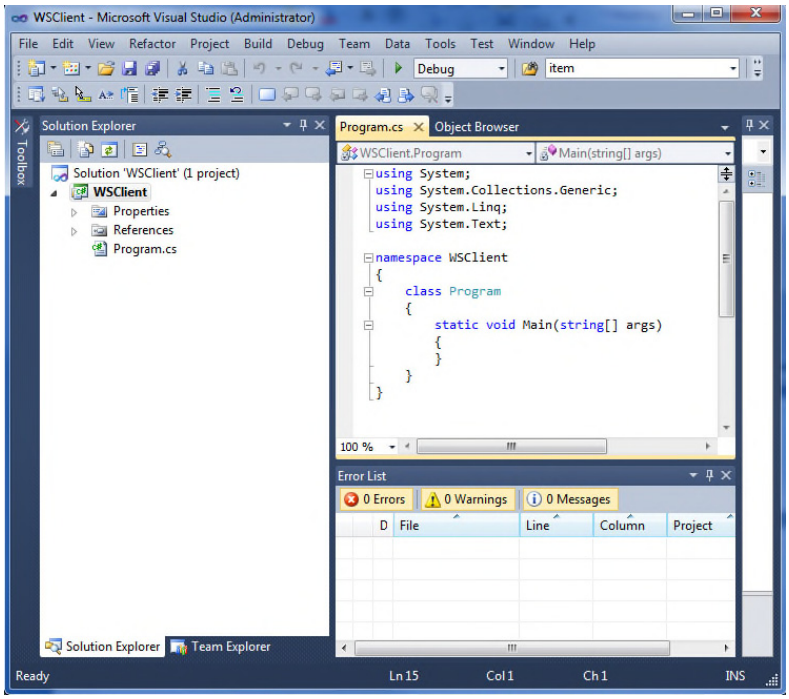
1. Open Visual Studio.

   Select **File** > **New** > **Project...** > **Console Application**



   Type in the name of your application (i.e. `WSClient`) and click **OK**.

2. On the **Solution Explorer** panel, right click on the `WSClient` project and select **Add Service Reference...**

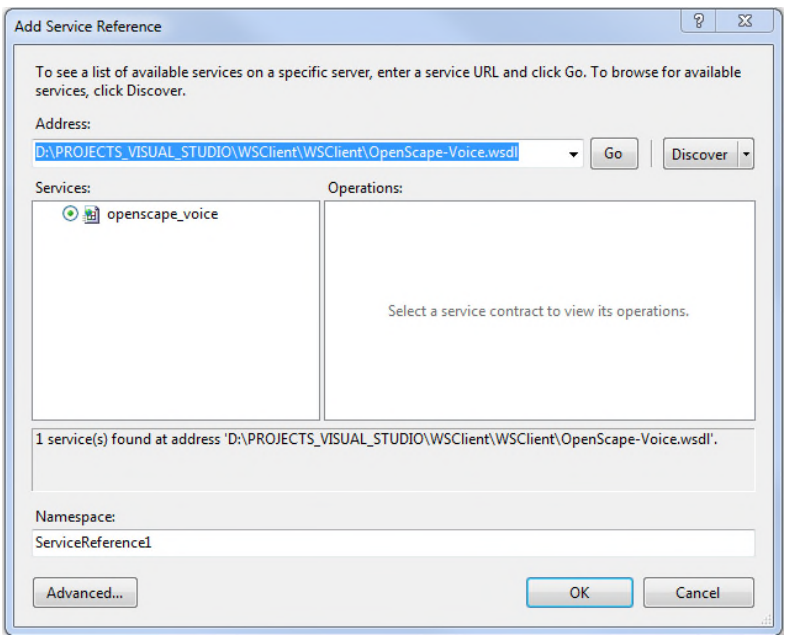


3. In the **Add Service Reference** screen:

   a) Fill in the path to the WSDL file on the **Address** field, and press **Go**.

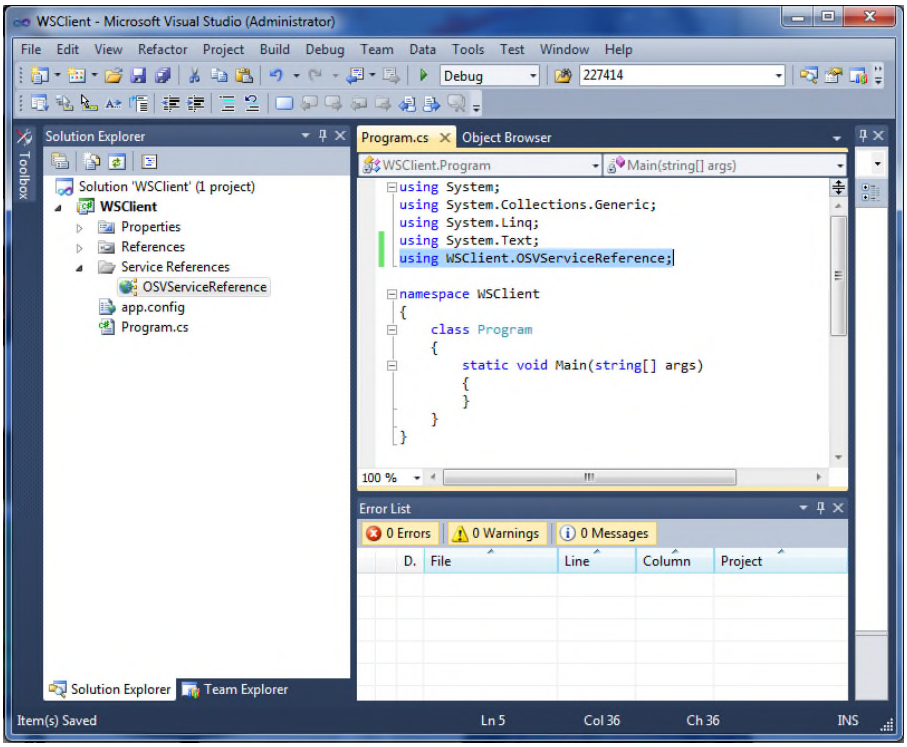
   `openscape_voice` service is now listed under the **Services** area.

   b) Change the **Namespace** to `OSVServiceReference`



   c) Click **OK**.

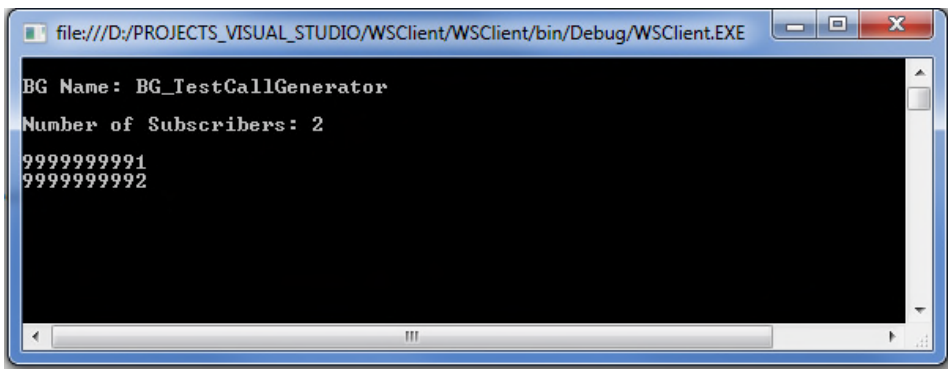
4. Add the `WSClient.OSVServiceReference` namespace on your program in order to use the service reference you created.



5. You are now ready to use the service reference (`OSVServiceReference`) to create a client. A code snippet is provided below that demonstrates the retrieval of the number of subscribers belonging to a specific Business Group (`BG_TestCallGenerator`).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using WSClient.OSVServiceReference;
namespace WSClient
{
    class Program
    {
        static void Main(string[] args)
        {
            System.ServiceModel.BasicHttpBinding binding = new System.ServiceModel.BasicHttpBinding();
            System.ServiceModel.EndpointAddress address = new System.ServiceModel.EndpointAddress("http://
            10.7.182.10:8767");

            openscape_voicePortTypeClient osv = new OSVServiceReference.openscape_voicePortTypeClient(binding,
            address);

            hiqGLOBALHEADER hiqHEADER = new OSVServiceReference.hiqGLOBALHEADER();
            hiqHEADER.OperatorId = Environment.UserName;

            GetSubscriberListRequest getSubListReq = new GetSubscriberListRequest();
            getSubListReq.BGName = "BG_TestCallGenerator";

            GetSubscriberListEntry[] GetSubscriberListEntryInfo;
            int TotalSubsFound = 0;
            PaginatedInfo PageInfo;
            ResultCodeStruct result;
            GetSubscriberList getSubscriberList = new GetSubscriberList();
            getSubscriberList.GetSubListReq = getSubListReq;

            GetSubscriberListResult rc = osv.GetSubscriberList(hiqHEADER, getSubscriberList);
            result = rc.Result;
            PageInfo = rc.PaginatedInfo;
            TotalSubsFound = rc.TotalSubsFound;
            GetSubscriberListEntryInfo = rc.GetSubscriberListData;

            Console.Write("\nBG Name: " + getSubListReq.BGName + "\n");
            Console.Write("\nNumber of Subscribers: " + TotalSubsFound + "\n\n");
            foreach (GetSubscriberListEntry item in GetSubscriberListEntryInfo)
            {
                Console.Write(item.ServiceId + "\n");
            }
        }
    }
}
```

6. Press **F5** to run your project. Check the output on the console.



Index