Avaya Aura® Contact Center Open Networking Web Services

Release 7.0.0

Issue 3.0

June 2016

Notice

While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

Documentation disclaimer

“Documentation” means information published by Avaya in varying mediums which may include product information, operating instructions and performance specifications that Avaya may generally make available to users of its products and Hosted Services. Documentation does not include marketing materials. Avaya shall not be responsible for any modifications, additions, or deletions to the original Published version of documentation unless such modifications, additions, or deletions were performed by Avaya. End User agrees to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation, to the extent made by End User.

Link disclaimer

Avaya is not responsible for the contents or reliability of any linked websites referenced within this site or documentation provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

Warranty

Avaya provides a limited warranty on Avaya hardware and software. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya’s standard warranty language, as well as information regarding support for this product while under warranty is available to Avaya customers and other parties through the Avaya Support website: http://support.avaya.com or such successor site as designated by Avaya. Please note that if you acquired the product(s) from an authorized Avaya Channel

Partner outside of the United States and Canada, the warranty is provided to you by said Avaya Channel Partner and not by Avaya.

Licenses

THE SOFTWARE LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE, HTTP://SUPPORT.AVAYA.COM/LICENSEINFO

OR SUCH SUCCESSOR SITE AS DESIGNATED BY AVAYA, ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AVAYA CHANNEL PARTNER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AVAYA CHANNEL PARTNER. UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA CHANNEL PARTNER; AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING OR SELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS “YOU” AND “END USER”), AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE

AVAYA AFFILIATE (“AVAYA”).

Avaya grants you a license within the scope of the license types described below, with the exception of Heritage Nortel Software, for which the scope of the license is detailed below. Where the order documentation does not expressly identify a license type, the applicable license will be a Designated System License. The applicable number of licenses and units of capacity for which the license is granted will be one (1), unless a different number of licenses or units of capacity is specified in the documentation or other materials available to you. “Designated Processor” means a single stand-alone computing device. “Server” means a Designated Processor that hosts a software application to be accessed by multiple users.

License type(s)

Named User License (NU). You may: (i) install and use the Software on a single Designated Processor or Server per authorized Named User (defined below); or (ii) install and use the Software on a Server so long as only authorized Named Users access and use the Software. “Named User”, means a user or device that has been expressly authorized by Avaya to access and use the Software. At Avaya’s sole discretion, a “Named User” may be, without limitation, designated by name, corporate function (e.g., webmaster or helpdesk), an e-mail or voice mail account in the name of a person or corporate function, or a directory entry in the administrative database utilized by the Software that permits one user to interface with the Software.

Copyright

Except where expressly stated otherwise, no use should be made of materials on this site, the Documentation, Software, Hosted Service, or hardware provided by Avaya. All content on this site, the documentation, Hosted Service, and the Product provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may

not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software unless expressly authorized by Avaya. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya can be a criminal, as well as a civil offense under the applicable law.

Third Party Components

“Third Party Components” mean certain software programs or portions thereof included in the Software or Hosted Service may contain software (including open source software) distributed under third party agreements (“Third Party Components”), which contain terms regarding the rights to use certain portions of the Software (“Third Party Terms”). As required, information regarding distributed Linux OS source code (for those Products that have distributed Linux OS source code) and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply is available in the Documentation or on Avaya’s website at:

http://support.avaya.com/Copyright or such successor site as designated by Avaya. You agree to the Third Party Terms for any such Third Party Components.

THIS PRODUCT IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO (i) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (ii) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE HTTP://WWW.MPEGLA.COM.

Note to Service Provider

The Product or Hosted Service may use Third Party Components subject to Third Party Terms that do not allow hosting and require a Service Provider to be independently licensed for such purpose. It is your responsibility to obtain such licensing.

Preventing Toll Fraud

"Toll Fraud" is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or is not working on your company's behalf). Be aware that there can be a risk of Toll Fraud associated with your system and that, if Toll Fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya Toll Fraud intervention

If you suspect that you are being victimized by Toll Fraud and you need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Support website: http://support.avaya.com or such successor site as designated by Avaya. Suspected security vulnerabilities with Avaya products should be reported to Avaya by sending mail to: securityalerts@avaya.com.

Trademarks

The trademarks, logos and service marks ("Marks") displayed in this site, the Documentation, Hosted Service(s), and Product(s) provided by Avaya are the registered or unregistered Marks of Avaya, its affiliates, or other third parties. Users are not permitted to use such Marks without prior written consent from Avaya or such third party which may own the Mark. Nothing contained in this site, the Documentation, Hosted Service(s) and Product(s) should be construed as granting, by implication, estoppel, or otherwise, any license or right in and to the Marks without the express written permission of Avaya or the applicable third party.

Avaya is a registered trademark of Avaya Inc.

All non-Avaya trademarks are the property of their respective owners. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

All non-Avaya trademarks are the property of their respective owners, and "Linux" is a registered trademark of Linus Torvalds.

Downloading Documentation

For the most current versions of Documentation, see the Avaya Support website: http://support.avaya.com or such successor site as designated by Avaya.

Contact Avaya Support

See the Avaya Support website: http://support.avaya.com for Product or Hosted Service notices and articles, or to report a problem with your Avaya Product or Hosted Service. For a list of support telephone numbers and contact addresses, go to the Avaya Support website: http://support.avaya.com (or such successor site as designated by Avaya), scroll to the bottom of the page, and select Contact Avaya Support.

Contents

Chapter 1: Revision History

Date	Revision #	Summary of Changes
December 2015	*Version 1.0*	Initial Avaya Aura® Contact Center Release 7.0
25 April 2016	*Version 2.0*	Updates for Avaya Aura® Contact Center 7.0 Release
13 June 2016	*Version 3.0*	Avaya Aura® Contact Center 7.0 Release version

Chapter 2: Overview

The Avaya Aura Contact Center (AACC) Open Networking Web Services enable third-party applications to transfer a call between different AACC nodes in a network. Any attached data associated with the call remains intact.

Third-party applications can reserve a Landing Pad on the target AACC node enabling the call to be transferred with data attached. The web services will also provide the functionality to cancel the reserving of a Landing Pad freeing it up for other calls to be transferred across the network.

This web service is hosted on the CCMS server of the target node and requires that the Universal Networking component is enabled on this CCMS server. This service relies on Universal Networking to be running and the required landing pads to be configured.

This web service can be optionally configured to use TLS. If this functionality is enabled the customer will be required to provide the necessary certificates. Refer to the Avaya Aura Contact Center documents for details about how to configure Web Services on the CCMS server.

When the service is configured using the default options the WSDL can be found at the following location, where the *<CCMS HostName>* is the host name of the CCMS server.

http(s)://*<CCMS HostName>*:9070/SOAOI/services/OpenNetworking?wsdl

This Web Service Definition Language (WSDL) is a machine readable description of the functionality being offered by this web service. Various technologies use this WSDL to interrogate the web service and create the relevant proxies to send and receive SOAP messages with the web service.

What is the SDK

The Avaya Aura® Contact Center Open Networking Web Services SDK can be downloaded to your client machine and contains information about how third-party applications can call the Open Networking Open Interfaces.

SDK contents

The SDK contains the following elements:

- Introduction and Tutorial PDF documentation
- Javadoc API documentation
- Installed client
- Source Java code for reference client implementations

SDK support

Support for the SDK APIs is supplied by through your Developer Partner Program.

Related Avaya Aura Contact Center documents

For more information about configuring Open Networking, https, security and certification, refer to the following Avaya Aura Contact Center documents:

- Avaya Aura® Contact Center Overview and Specification
- Avaya Aura® Contact Center commissioning for Avaya Aura® Unified Communications
- Avaya Aura® Contact Center Commissioning for Avaya Communication Server 1000
- Avaya Aura® Contact Center Server Administration

Chapter 3: Open Networking Introduction

Third-party applications can reserve a Landing Pad on the AACC target node enabling the call to be transferred with data attached. The web services will also provide the functionality to cancel the reserving of a Landing Pad freeing it up for other calls to be transferred across the network.

Typically a third-party application will follow these steps when utilizing this web service.

1. Reserve Landing Pad – Reserve a landing pad on the target node for transferring the call.
2. Cancel Landing Pad – Cancel the reservation of the landing pad for a networked call.

Some useful definitions

Term	Definition
Contact Center contact	A contact represents a media agnostic session with and agent. A contact can represent many different contact types such as voice, video, email, im etc. Before a contact is created a third party application has the option of associating their own identifier (externalId) with the contact. This identifier must be unique with respect to the system and can be used to identify\retrieve the contact at a later stage.
Intrinsics	Intrinsics are a series of key value pairs that can be associated with a Contact. Routing of the contact can be influenced at contact creation time by populating contact intrinsics which can be used by the scripting engine to make routing decisions. Alternatively these intrinsics can be used to convey application specific information along with the call such as a customer specific Id, name, url etc. When the call alerts at the agents desktop this information can be retrieved from the contact.

Reserve Landing Pad

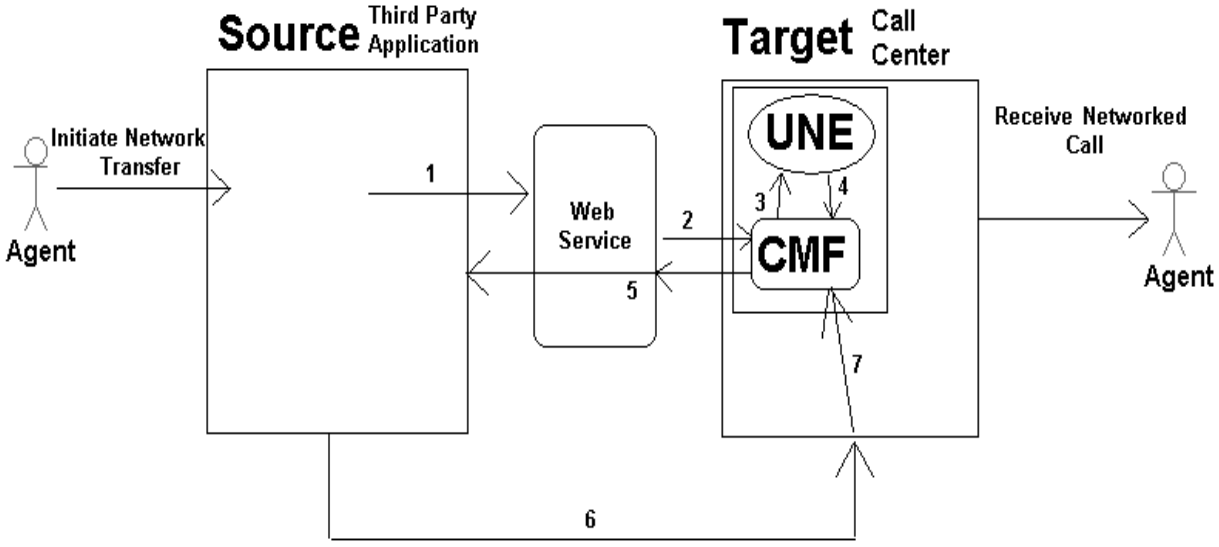
Before a third-party application can transfer a contact, it must first reserve a landing pad on the target node. To initiate a transfer, an application must submit a *reserveCDNLandingPad* request to the Open Networking Open Interfaces.

The *reserveCDNLandingPad* requires the following parameters:

- externalContactId – a unique id for third-party applications to identify and retrieve the newly created contact using either the *getOQContact* or the *dropOQContact.*
- *destinationAddress* – the final destination address that the call will arrive at.
- *contactData* – data associated with the *Contact* when the call arrives at the final destination.
- intrinsics– an array of key value pairs that are associated with the *Contact*.
- ReasonCode (required)– the reason for the landing pad to be reserved Reason.TRANSFER_OPEN_NET_INIT or Reason.CONFERENCE_OPEN_NET_INIT
- *contactGUID* (required) – a parameter containing the *Contact* GUID that will be used to uniquely identify the *Contact* at the target node. This GUID must be unique for all Contacts at the target node. This GUID will be returned in the response for this service call.
- authentication – The authentication object is used to validate the transaction, it contains the user credentials. Username, Domain and Password are the fields on the object.
- *landingPadName* –used to pass back the value of the LandingPad that has been reserved.

When this request is received on the target note an event will trigger UNE to create a logical contact to store the call's data and the service returns the reserved *Landing Pad Address* to the application. The application will then use this returned address to route a contact to the target node to be processed at that node.

For this process to work the third party must know the location of the target node that it wishes to transfer the call to, the Third Party will need to process the destination address appropriately and will also need to manipulate the returned address so the call can be routed correctly.



Web Service Networking Process

Cancel Landing Pad

When a third-party application has completed the transfer, the application must cancel the landing pad to make the landing pad available for subsequent calls.

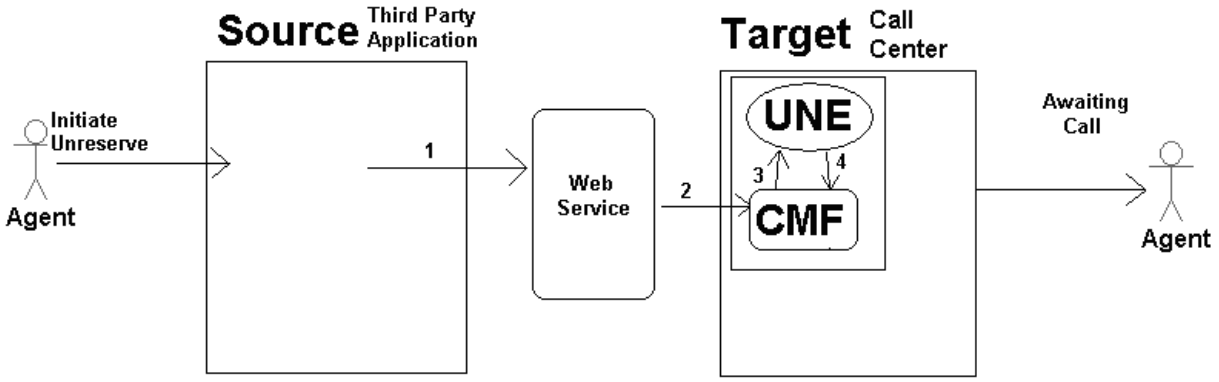
To initiate a cancel request an application will submit a *cancelCDNLandingPad* request to the Open Networking Open Interfaces.

The *cancelCDNLanding* requires the following parameters:

- *landingPadName* - the name of the landing pad that is to be unreserved.
- authentication – The authentication object which contains the user credentials.

When the target node receives the cancel request it will trigger an event in UNE that will unreserved the *Landing Pad Address* and remove any contacts associated with the landing pad. The landing pad is now available for other network calls.

For this process to work the third-party application must know the location of the target node at which it wishes to un-reserve the Landing Pad.



Web Service Unreserving Landing Pad Process

Chapter 4: Tutorial

This tutorial describes how to validate a third-party application using the Open Networking web services by reserving a landing pad CDN and then cancelling that reservation using Java technology.

It is assumed the developer has some experience with the Java programming language and the Eclipse Integrated Development Environment (IDE).

Assumptions about this tutorial:

- The WSDL for Open Networking is accessible from the client development machine.
- The IDE referenced and the screen shots used were all taken from Eclipse.
- For simplicity, the example does not use secure web services.
- Common programming practices such as controls, events, threading, and exceptions are left to the individual programmer and are not covered in this tutorial.
- Exception handling is not shown in the code samples for brevity.

Prerequisites

This tutorial assumes that UNE is installed and configured and that the user also has a license for this Open Interfaces Open Networking web services.

Using the SDK reference client sample code

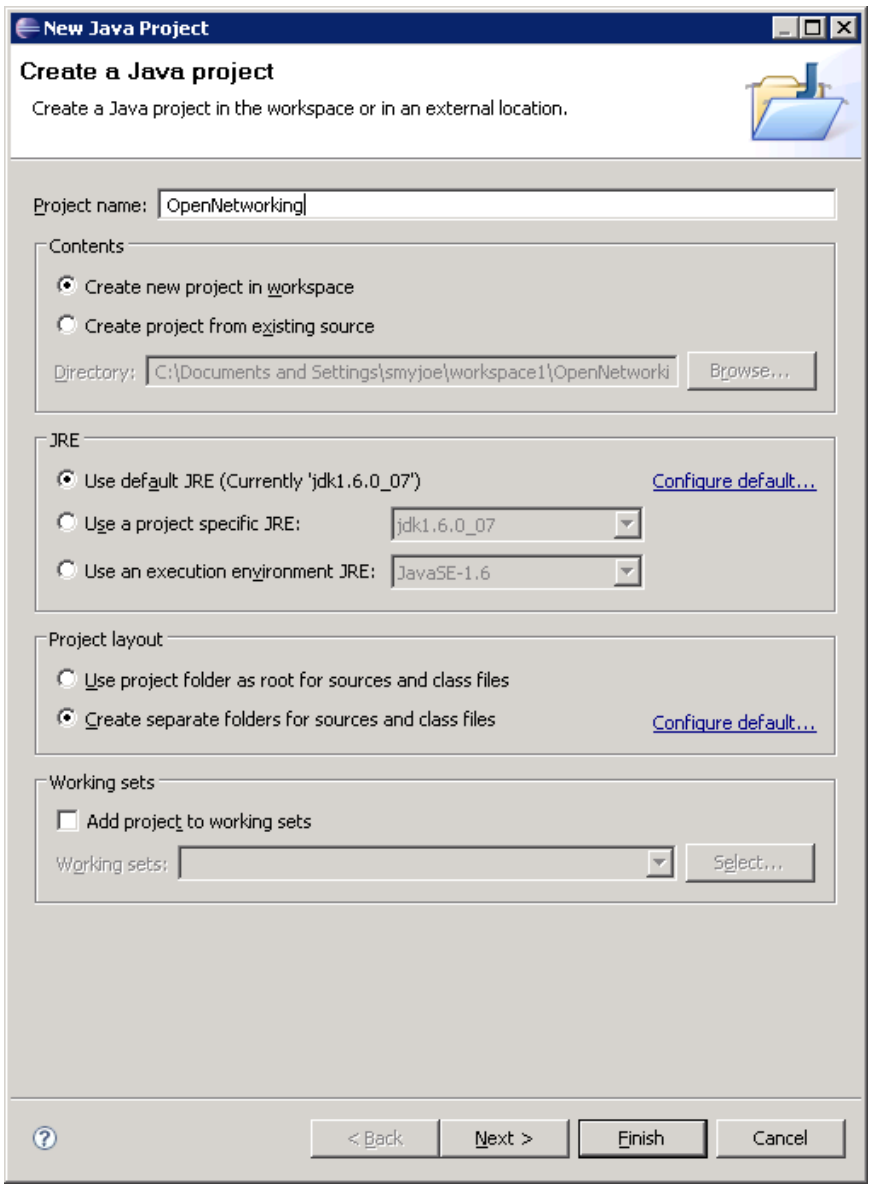
Follow these steps to use the SDK reference client sample code:

1. Create an Eclipse project
2. Import the WSDL into an eclipse project
3. Create a test client
4. Reserve a Landing Pad
5. Using the complete listing

Create an Eclipse Project

1. Open the Eclipse IDE and select File\New from the menu to create a new java project.
2. Enter the project name as *OpenNetworking*.
3. Select finish to create the project, a new project will appear in the Package Explorer of the eclipse environment.

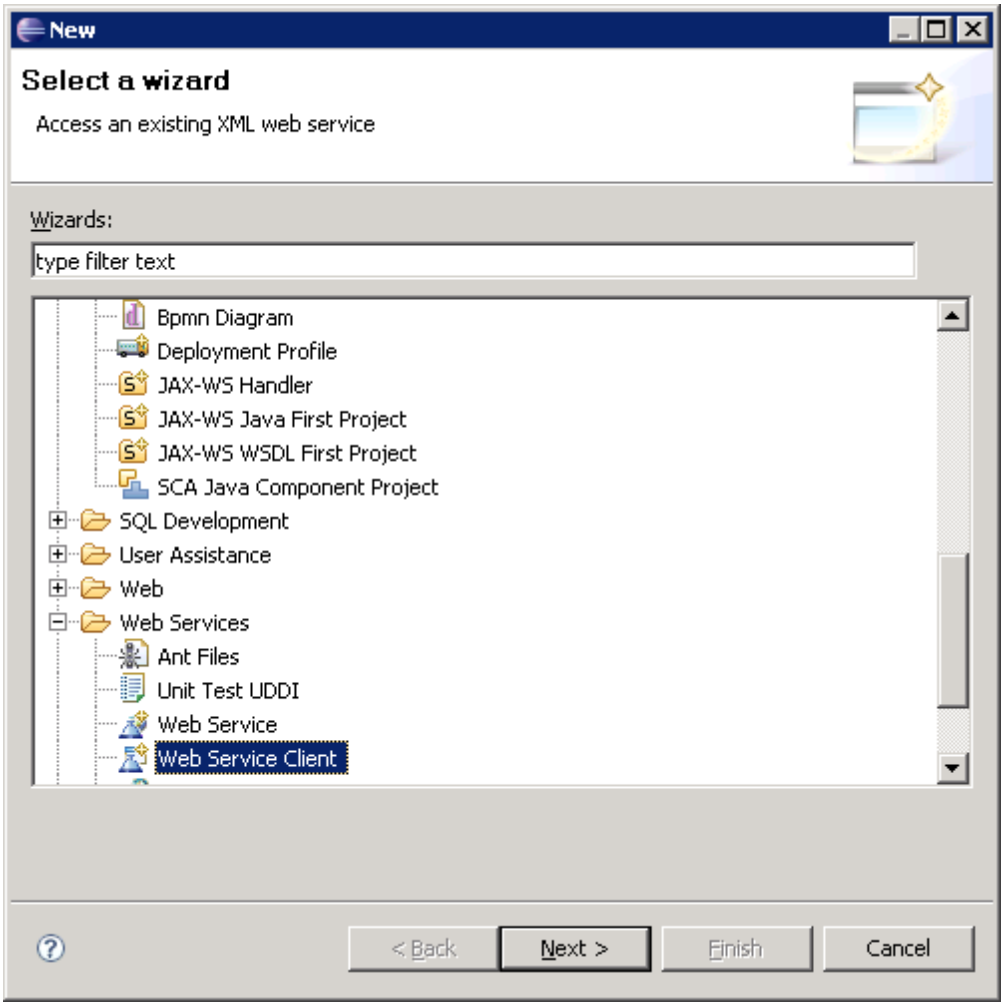
   Note:
   Full source for the complete sample is available as part of the SDK at:
   `/Open Interfaces Open Networking SDK/reference client`



Import the WSDL into an eclipse project

1. The next step will be to create a Web Service client for our WSDL. This will create the relevant proxy objects for our service as well as updating the project classpath.

2. Select the project and select New\Other from the drop down menu. The following dialog appears.



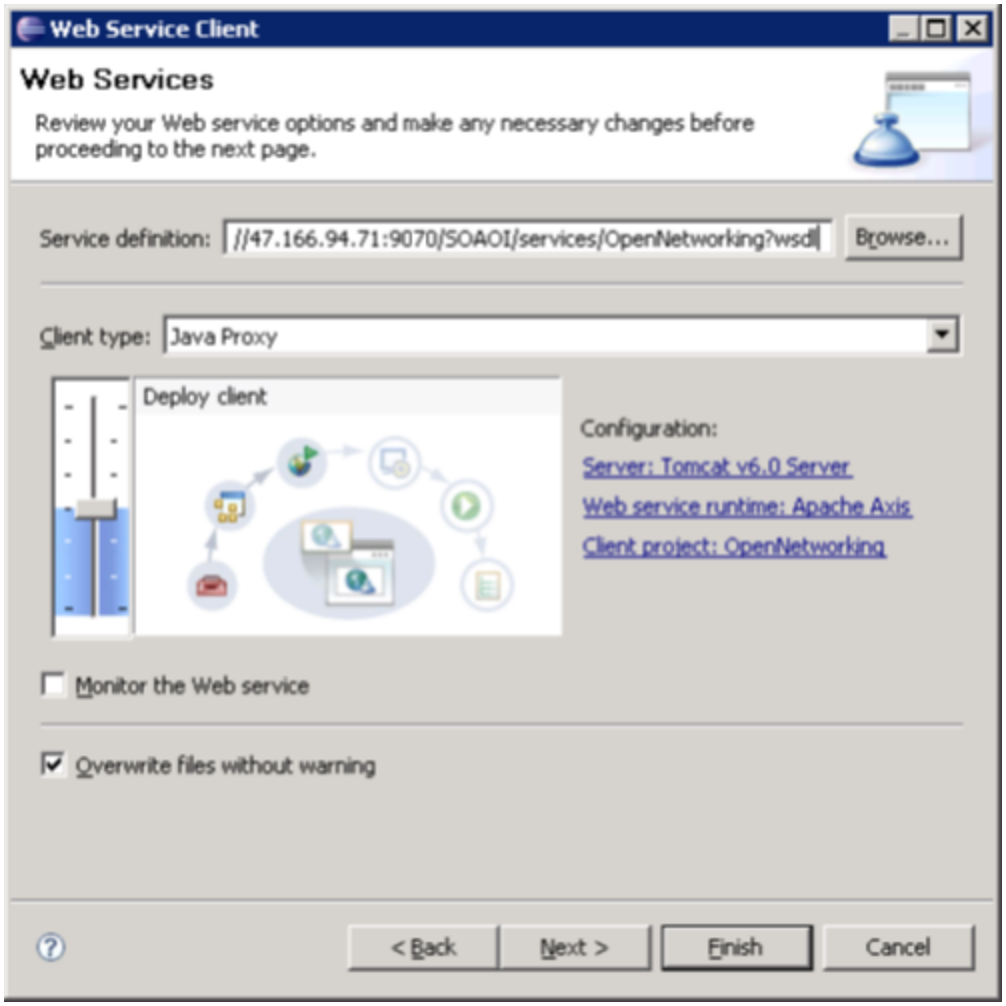
New Web Service Client

3. Select the *Next* button to open the Web Service Client dialog. Enter in the location of the WSDL in the service definition text box. The WSDL can be found at the following location, where <CCMS Server> is the name of the CCMS server.

   `http://<CCMS Server>:9070/SOAOI/services/OpenNetworking?wsdl`
   `or`
   `https://<CCMS Server FQDN>:9070/SOAOI/services/OpenNetworking?wsdl`

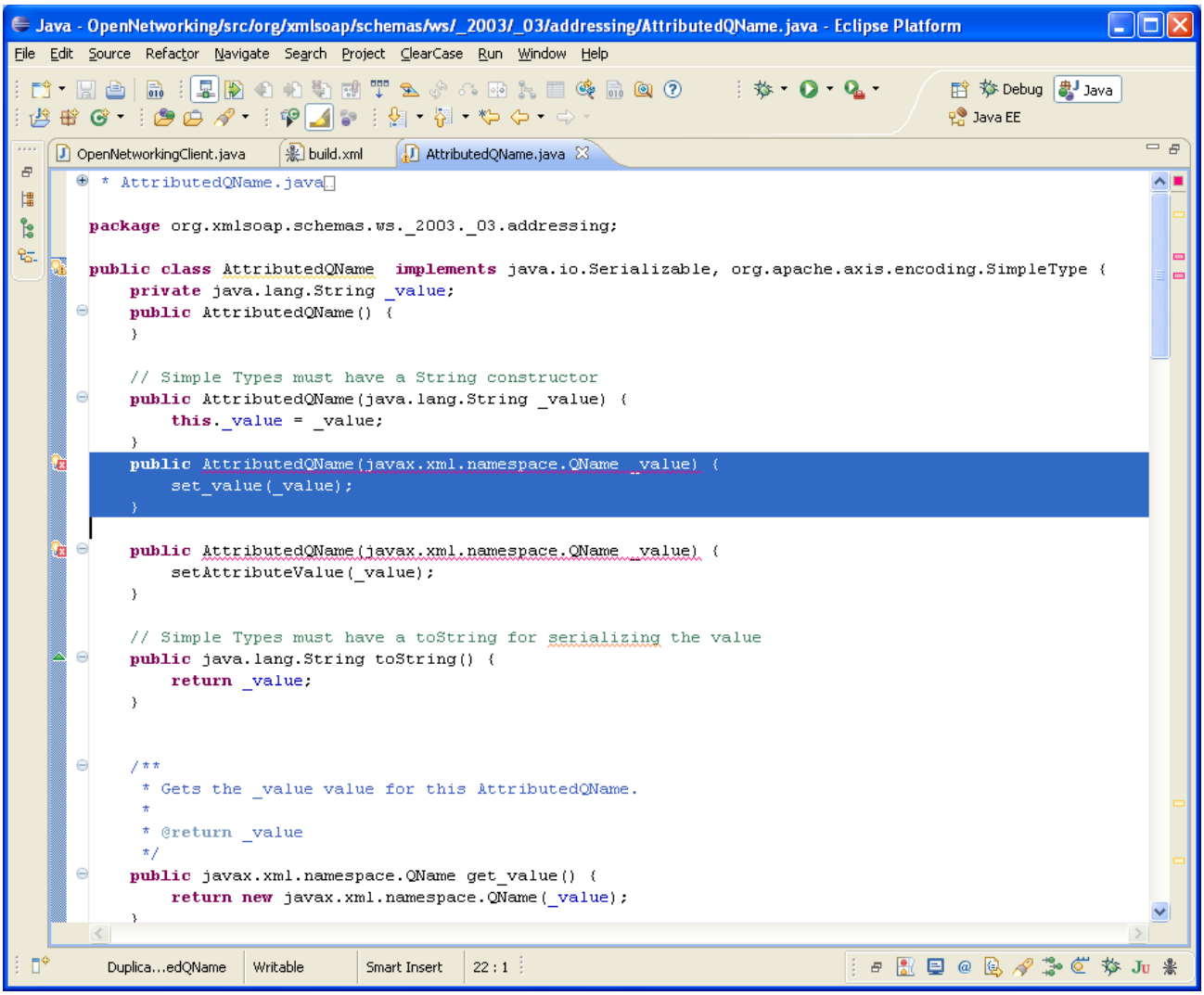
   Note: the hostname refers to the CCMS server hosting the Open Networking Open Interface. Depending on the customers configuration the port number – 9070 – may differ than that used in the example

4. Once the WSDL location has been entered hit the *Finish* button and the relevant proxies will be generated. Note this might take a few minutes depending on the machine and network speed.



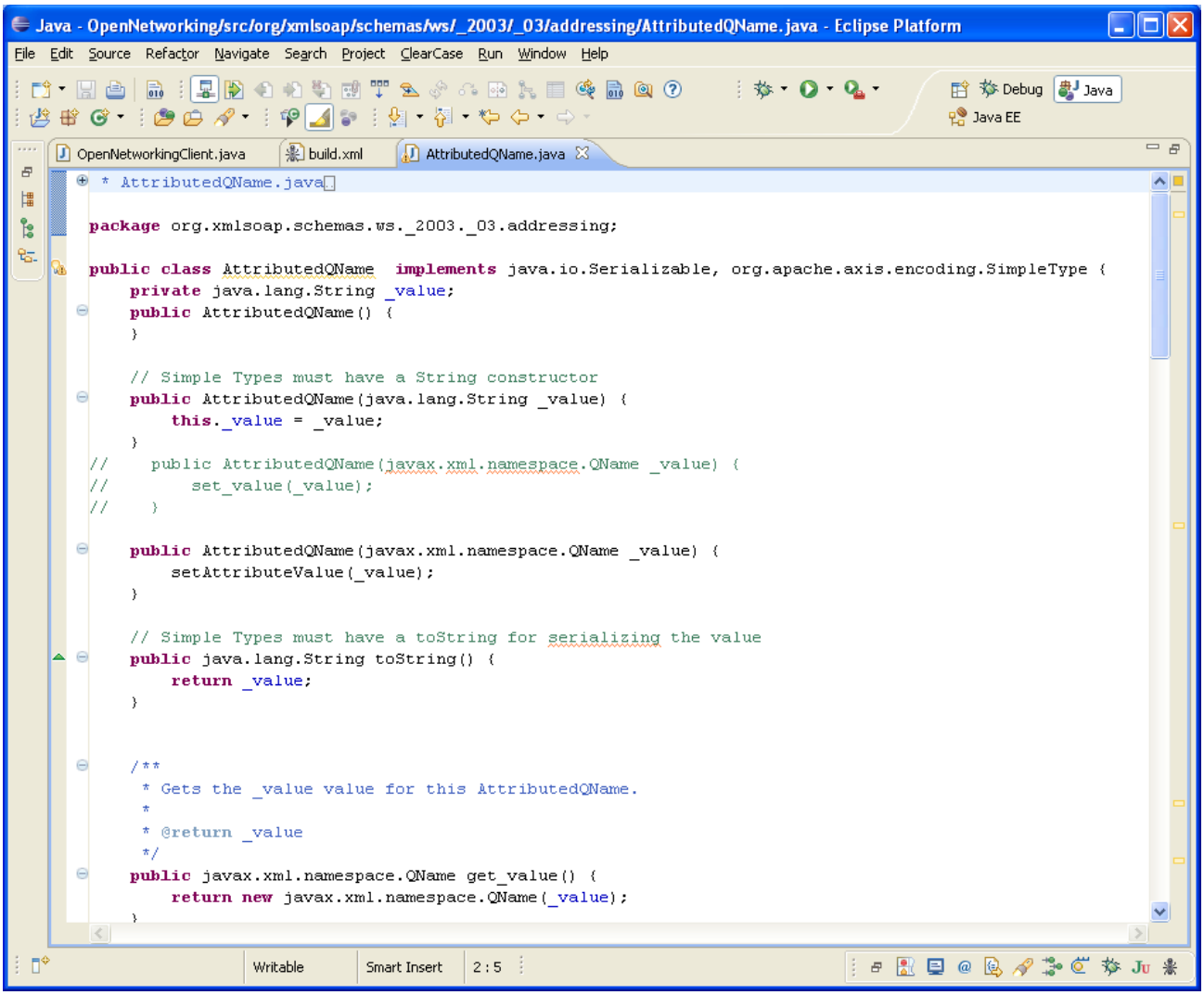
Web Service Client Wizard

5. When the wizard has completed the following code will have been generated.



Generated Proxies

6. An error in the proxy generation tool cause a constructor to be created for QName. Please comment out the constructor that contains the set_Value to remove the error. See next screen shot.



Generated Proxy with Error removed.

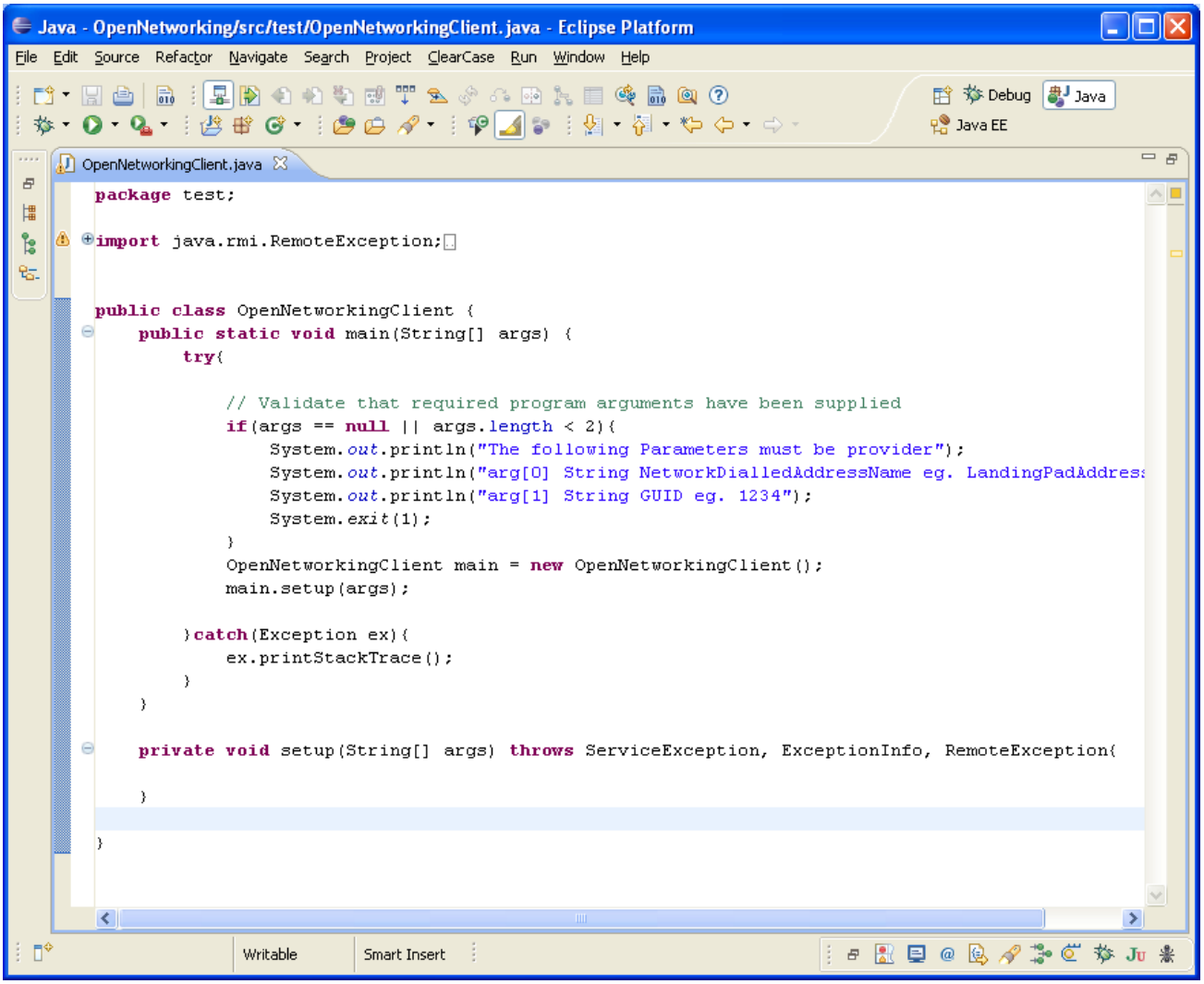
Create a test client

The next step will be to create a test client. Select the project and from the popup menu select New/Class to create the class *OpenNetworkingClient* in the package *test*. The following screen shot shows the skeleton of the class.

The main method expects two inputs string parameters:

- A networked dialed address to provide the address of the target CDN
- A GUID to identify the contact being created on the target node

The class also includes an empty setup( ) method which will be populated in the following steps.



OpenNetworkingClient

Reserve a Landing Pad

The next step will be to call the reserveCDNLandingPad( ) web service operation. This method has the following parameters.

Required parameters are marked (R) and optional parameters are marked (O).

- *contactGUID* (R) – a parameter containing the *Contact* GUID that will be used to uniquely identify the *Contact* at the target node. This GUID must be unique for all Contacts at the target node. This GUID will be returned in the response for this service call.
- *destinationCDN* (R) – the destination CDN for the contact.
- *contactData* (O) – data to be associated with the *Contact* when the call arrives at the final destination.
- intrinsics (O) – an array of key value pairs that are associated with the *Contact*.
- ReasonCode (R) – the reason for the landing pad to be reserved. It takes on of the following two values:
  Reason.TRANSFER_OPEN_NET_INIT or
  Reason.CONFERENCE_OPEN_NET_INIT
- Authentication (R) – the authentication object is used to validate the request, it contains the user credentials: username, and password and domain.
  Username is fixed to *OpenWsUser.* The default password is *Password123*.
  It's possible to change this via the CCMS Server Configuration tool. The default value for domain is *localhost*.

The *destinationCDN* is created during UNE setup and hence will change from customer to customer.

The following code snippet demonstrates how to call the reserve landing pad method.

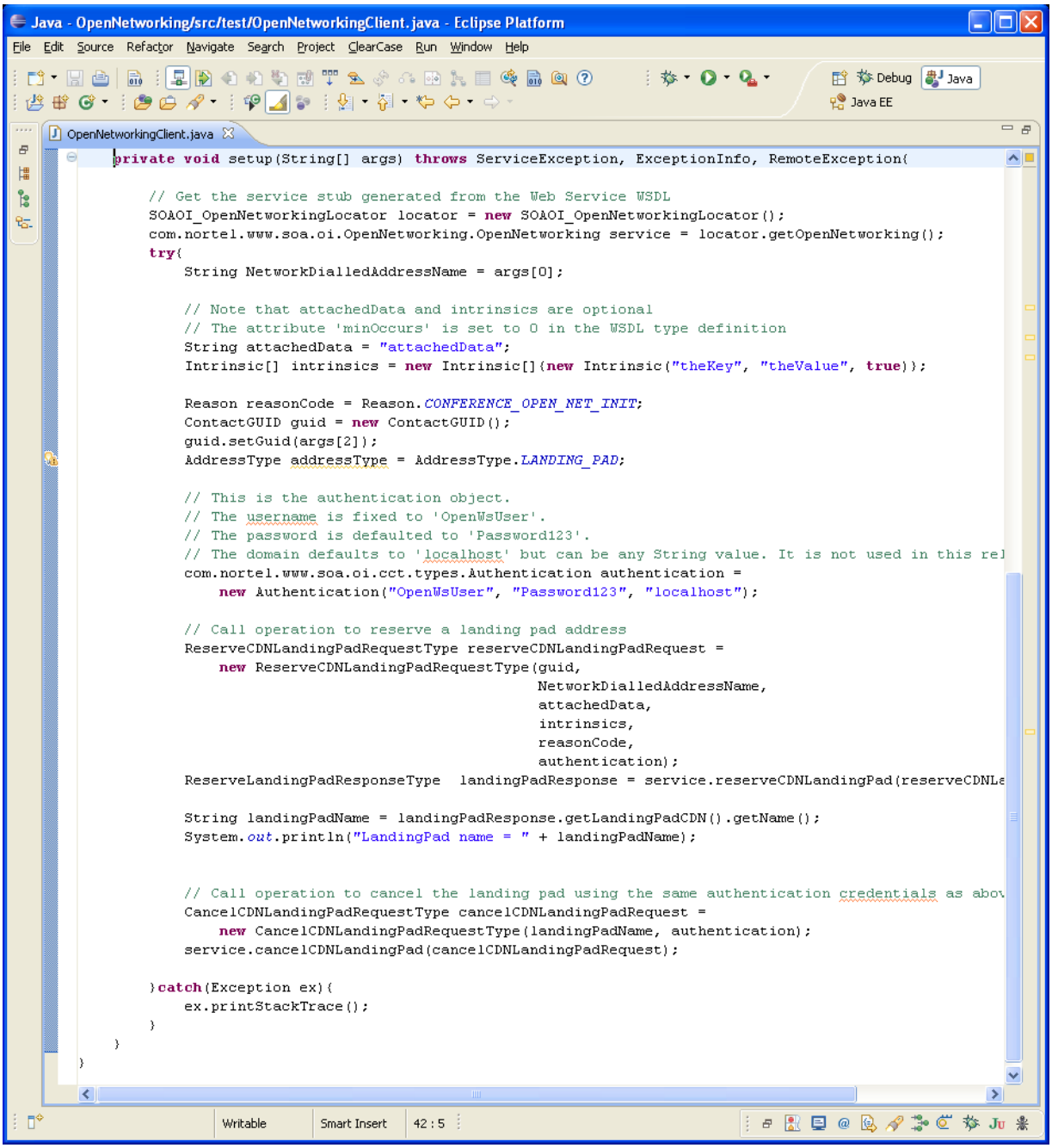
```
String attachedData = "attachedData";
Intrinsic[] intrinsics = new Intrinsic[]{new Intrinsic("theKey",
"theValue", true)};

Reason reasonCode = Reason.CONFERENCE_OPEN_NET_INIT;
ContactGUID guid = new ContactGUID();
guid.setGuid(args[2]);
AddressType addressType = AddressType.LANDING_PAD;
// This is the authentication object.
// The username is fixed to 'OpenWsUser'.
// The password is defaulted to 'Password123'.
// The domain defaults to 'localhost' but can be any String value. It
is not used in this release.

com.nortel.www.soa.oi.cct.types.Authentication authentication =
      new Authentication("OpenWsUser", "Password123", "localhost");

// Call operation to reserve a landing pad address
ReserveCDNLandingPadRequestType reserveCDNLandingPadRequest =
      new ReserveCDNLandingPadRequestType(guid,
            NetworkDialledAddressName,
      attachedData,
      intrinsics,
      reasonCode,
      authentication);

ReserveLandingPadResponseType landingPadResponse =
      service.reserveCDNLandingPad(reserveCDNLandingPadRequest);

String landingPadName =
      landingPadResponse.getLandingPadCDN().getName();

// Call operation to cancel the landing pad using the same
authentication credentials as above.
CancelCDNLandingPadRequestType cancelCDNLandingPadRequest =

new CancelCDNLandingPadRequestType(landingPadName, authentication);
service.cancelCDNLandingPad(cancelCDNLandingPadRequest);
```

Using the complete listing

The following is a complete listing of all the client application. The resultant project can be found in the accompanying SDK package, in the “reference client” folder.

```
package test;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import
org.oasis_open.docs.wsrf._2004._06.wsrf_WS_BaseFaults_1_2_draft_01_xsd.ExceptionInfo;

import com.nortel.www.soa.oi.OpenNetworking.SOAOI_OpenNetworkingLocator;
import com.nortel.www.soa.oi.OpenNetworking.types.CancelCDNLandingPadRequestType;
import com.nortel.www.soa.oi.OpenNetworking.types.ReserveCDNLandingPadRequestType;
import com.nortel.www.soa.oi.OpenNetworking.types.ReserveLandingPadResponseType;
import com.nortel.www.soa.oi.cct.types.AddressType;
import com.nortel.www.soa.oi.cct.types.Authentication;
import com.nortel.www.soa.oi.cct.types.AuthenticationLevel;
import com.nortel.www.soa.oi.cct.types.ContactGUID;
import com.nortel.www.soa.oi.cct.types.Intrinsic;
import com.nortel.www.soa.oi.cct.types.LandingPad;
import com.nortel.www.soa.oi.cct.types.Reason;
import com.nortel.www.soa.oi.cct.types.holders.ContactGUIDHolder;

public class OpenNetworkingClient {
	public static void main(String[] args) {
	try{
		// Validate that required program arguments have been supplied
		if(args == null || args.length < 2){
			System.out.println("The following Parameters must be provider");
			System.out.println("arg[0] String NetworkDialledAddressName eg.
			LandingPadAddressName");
			System.out.println("arg[1] String GUID eg. 1234");
			System.exit(1);
			}
		OpenNetworkingClient main = new OpenNetworkingClient();
		main.setup(args);
		}catch(Exception ex){
			ex.printStackTrace();
		}

	}
private void setup(String[] args)
	throws ServiceException, ExceptionInfo, RemoteException{

	// Get the service stub generated from the Web Service WSDL
	SOAOI_OpenNetworkingLocator locator = new SOAOI_OpenNetworkingLocator();
	com.nortel.www.soa.oi.OpenNetworking.OpenNetworking service =
			locator.getOpenNetworking();

	try{
		String NetworkDialledAddressName = args[0];
		// Note that attachedData and intrinsics are optional
		// The attribute 'minOccurs' is set to 0 in the WSDL type
		definition
		String attachedData = "attachedData";
		Intrinsic[] intrinsics =
		new Intrinsic[]{new Intrinsic("theKey", "theValue", true)};
```

```
                Reason reasonCode = Reason.CONFERENCE_OPEN_NET_INIT;
                ContactGUID guid = new ContactGUID();
                guid.setGuid(args[2]);
                AddressType addressType = AddressType.LANDING_PAD;

                // This is the authentication object.
                // The username is fixed to 'OpenWsUser'.
                // The password is defaulted to 'Password123'.
                // The domain defaults to 'localhost' but can be any String
                // value. It is not used in this release.
                com.nortel.www.soa.oi.cct.types.Authentication authentication =
                new Authentication("OpenWsUser", "Password123", "localhost");

                // Call operation to reserve a landing pad address
                ReserveCDNLandingPadRequestType reserveCDNLandingPadRequest =
                       new ReserveCDNLandingPadRequestType(guid,
                       NetworkDialledAddressName,
                       attachedData,
                       intrinsics,
                       reasonCode,
                       authentication);

                ReserveLandingPadResponseType landingPadResponse =
                service.reserveCDNLandingPad(reserveCDNLandingPadRequest);

                String landingPadName =
                landingPadResponse.getLandingPadCDN().getName();

                // Call operation to cancel the landing pad
                //using the same authentication credentials as above.
                CancelCDNLandingPadRequestType cancelCDNLandingPadRequest =
                new CancelCDNLandingPadRequestType(landingPadName, authentication);
                service.cancelCDNLandingPad(cancelCDNLandingPadRequest);
                }catch(Exception ex){
                ex.printStackTrace();
                }
        }
}
```

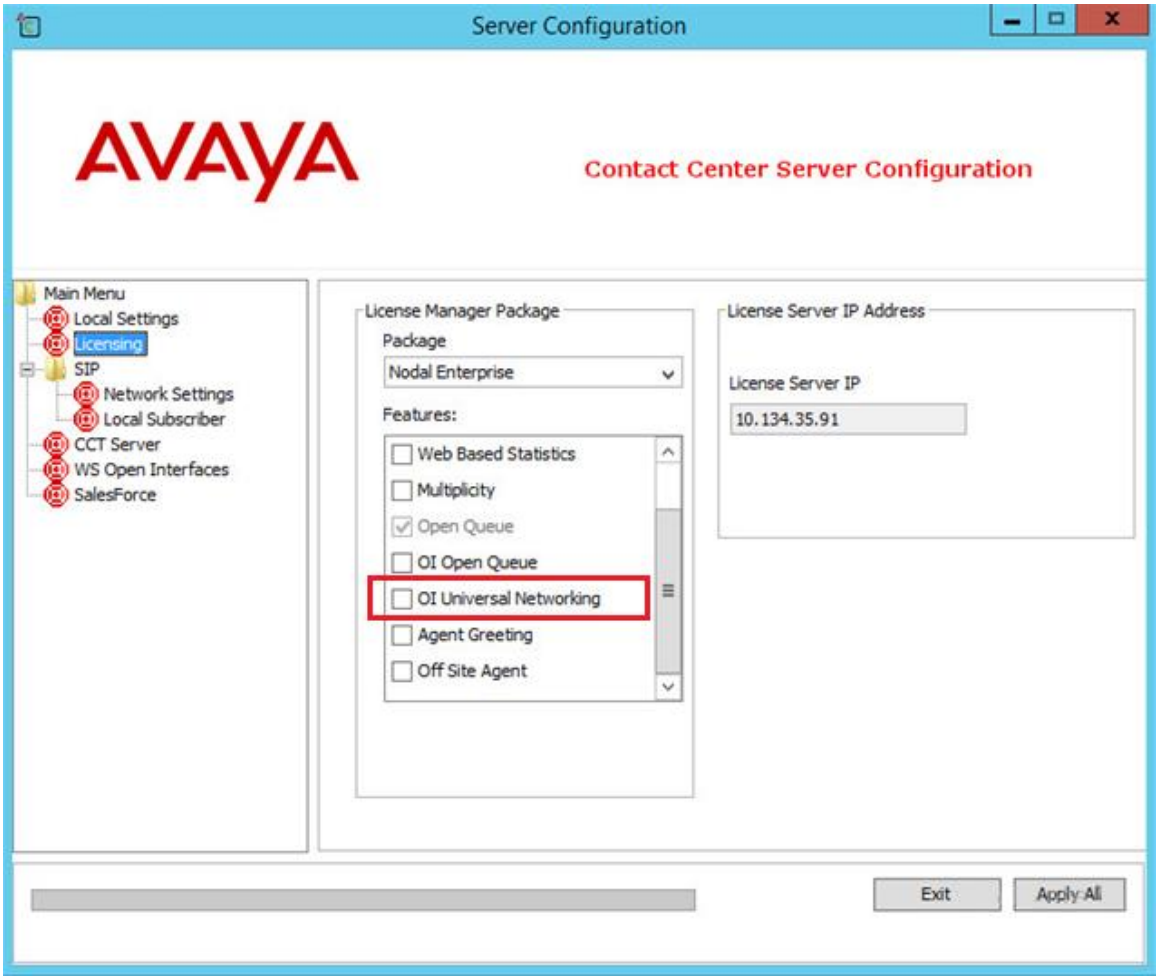


Code for the setup( ) Method

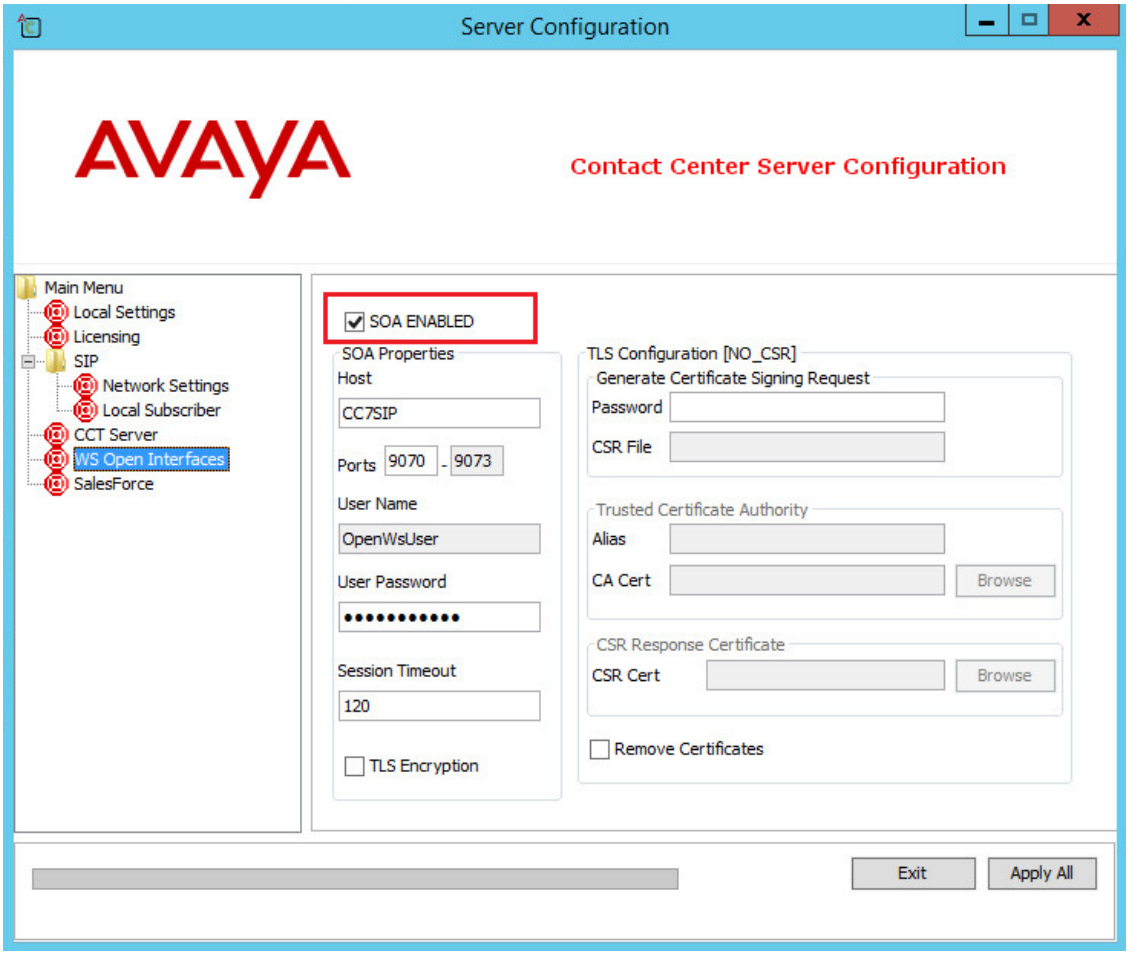
Chapter 5: Troubleshooting

Cannot access the Open Networking WSDL

1. From the **Licensing** tab under the Server Configuration dialog on the CCMS server, verify that **OI Universal Networking** is selected and that there is a valid license for these features.



2. From the **WS Open Interface** tab under the Server Configuration dialog, ensure that the SOA service is enabled and if TLS is enabled then ensure that you are using *https* in the URL as opposed to *http.*



3. Ensure that there is not conflict in the ports being used by the web service. The default range is 9070-9073.

4. If the service is visible on the CCMS server and not remotely, ensure that there is no firewall restricting access to the CCMS server.

LAST PAGE