Open Interfaces Open Networking Tutorial

This tutorial describes how to validate a 3rd Party Application using the Open Interfaces Open Networking web service by reserving a landing pad CDN and then cancelling that reservation using Java technology.

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

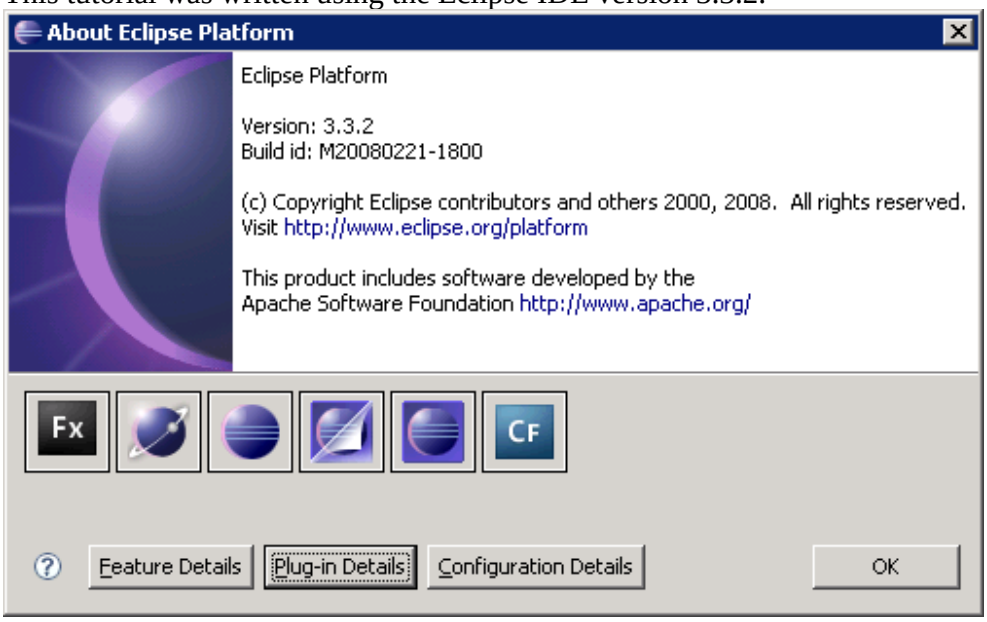
Assumptions about this tutorial:

- The WSDL definition of the Open Networking Open Interface is accessible from the client development machine.
- The Open Networking Open Interface WSDL definition was based on Contact Center 7.0 SU_02 release.
- The IDE referenced and the screen shots used were all take from Eclipse 3.3.2 on Windows 2003.
- 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.
- All code should be tested and reviewed before running against a production system.

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.

This tutorial was written using the Eclipse IDE version 3.3.2.



Tutorial

The following steps detail how:

1. Crate an Eclipse project
2. Import the WSDL into an eclipse project,
3. Call the reserveCDNLandingPad web service operation
4. Call the cancelCDNLandingPad web service operation

Step 1 - Create 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

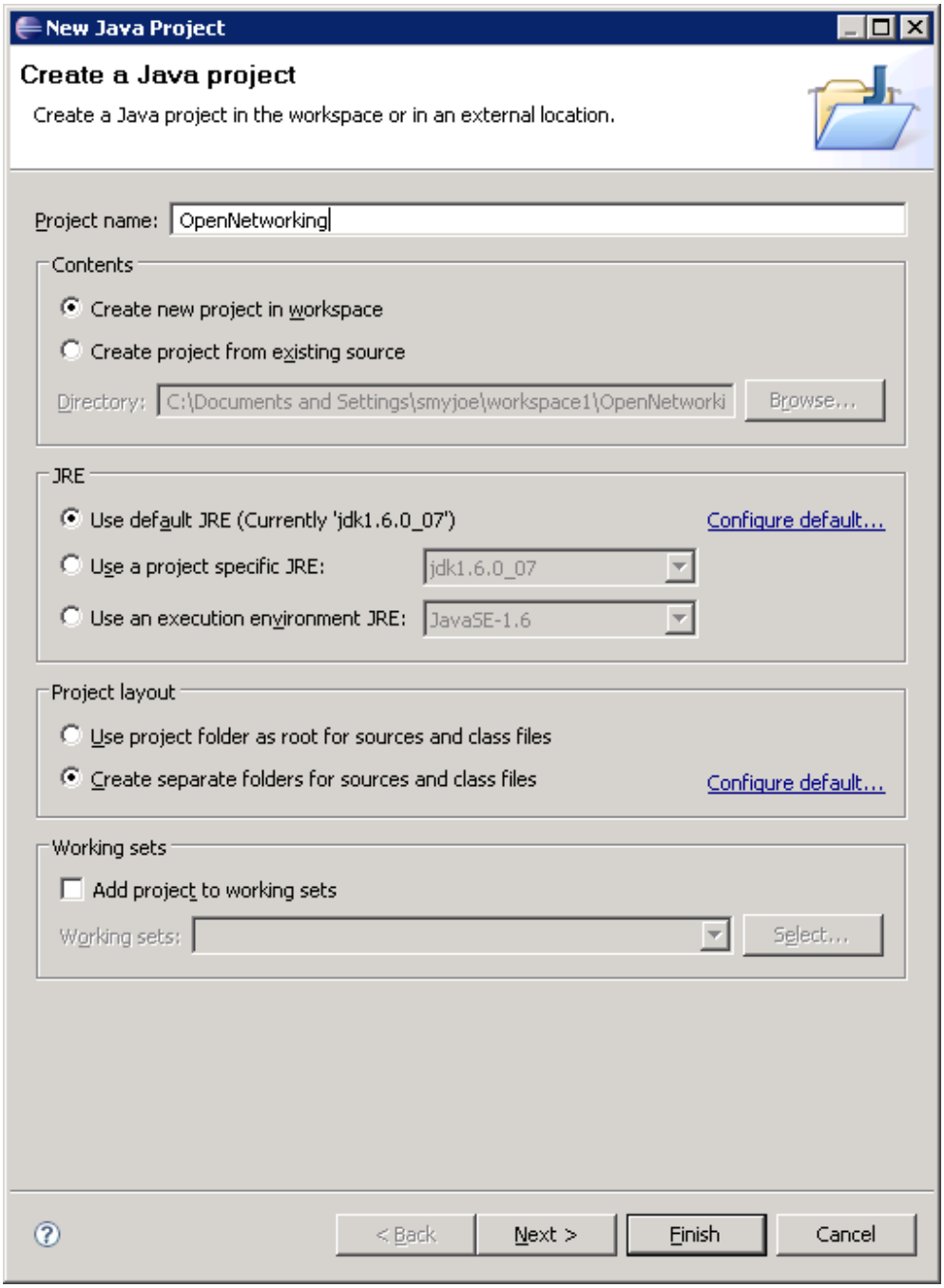


Figure 1 Create Project

Step 2 - Import WSDL

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 will appear.

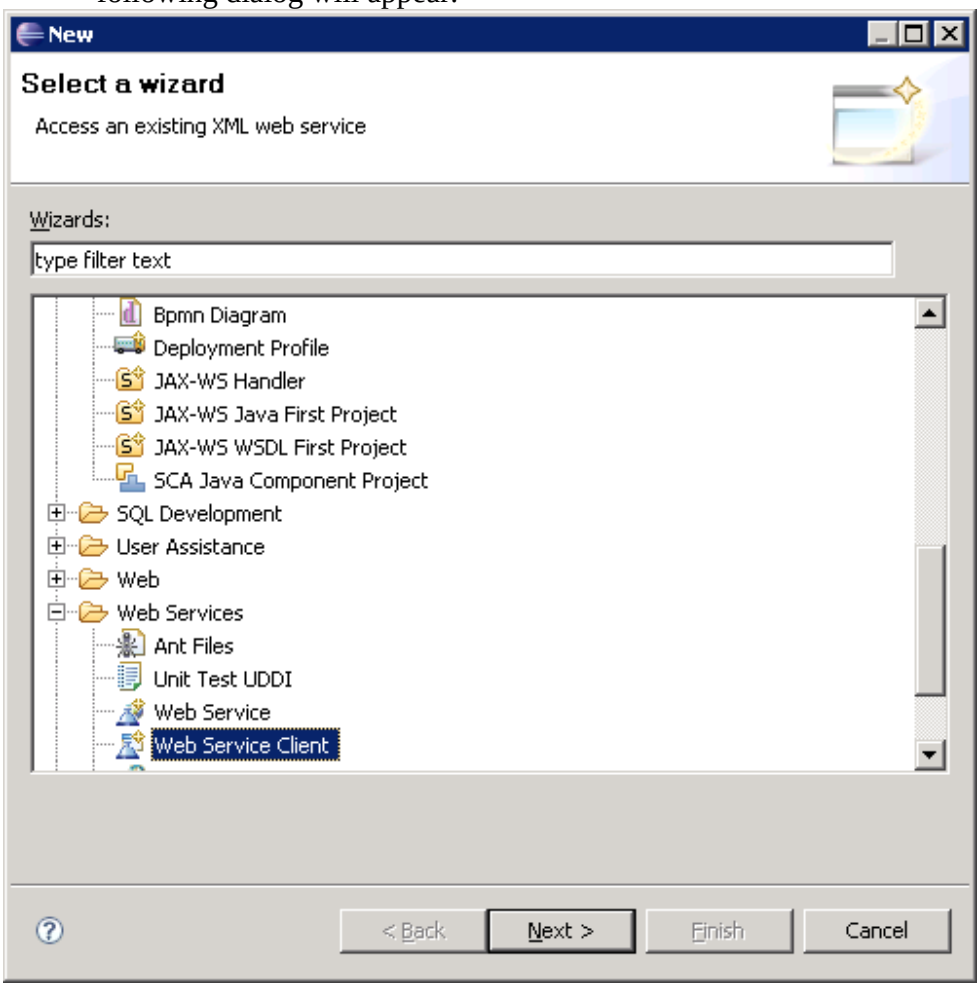


Figure 2 New Web Service Client

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>:9080/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 – 9080 – 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.

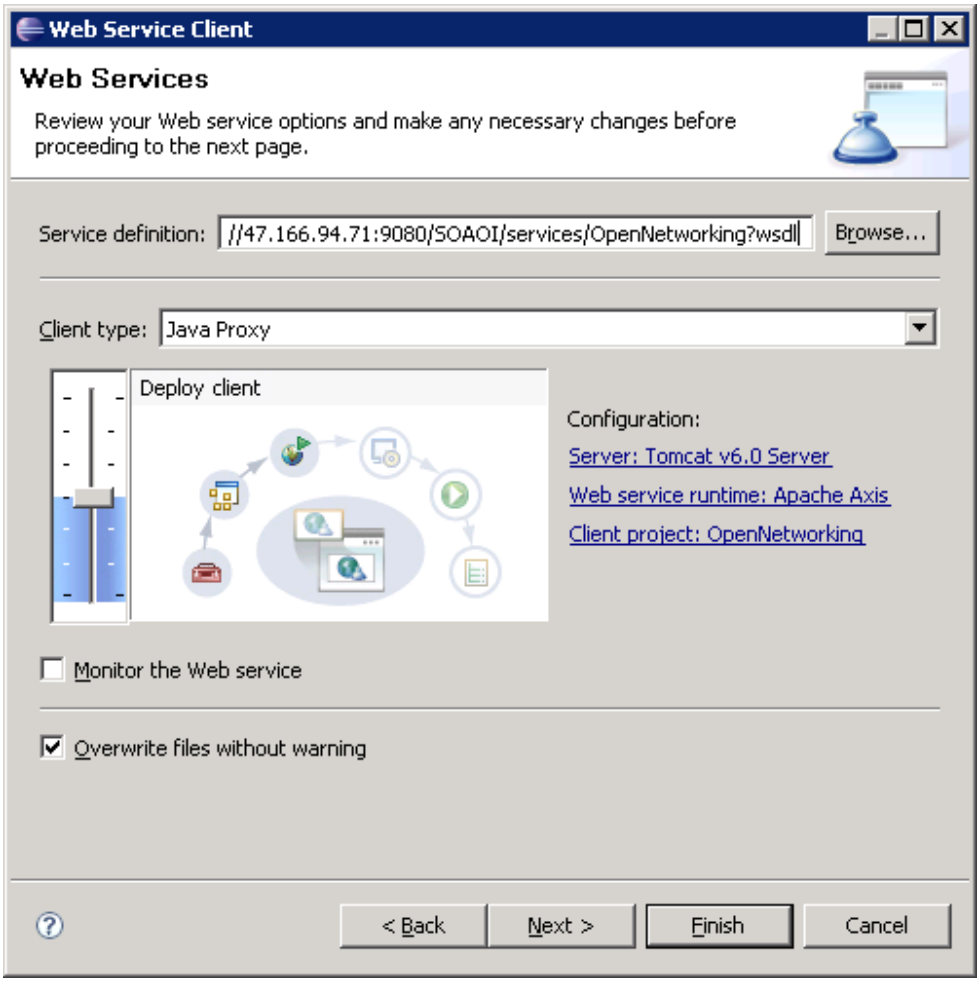


Figure 3 Web Service Client Wizard

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

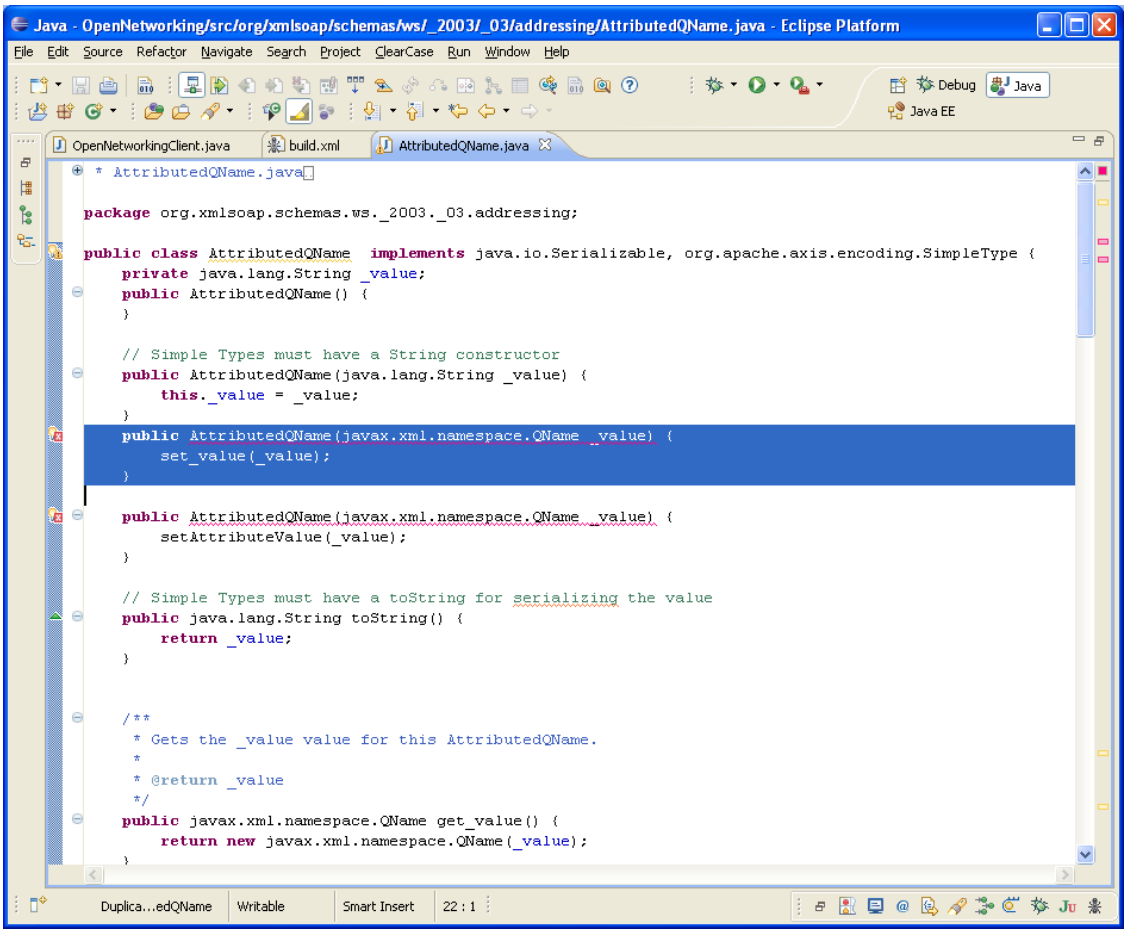


Figure 4 Generated Proxies

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

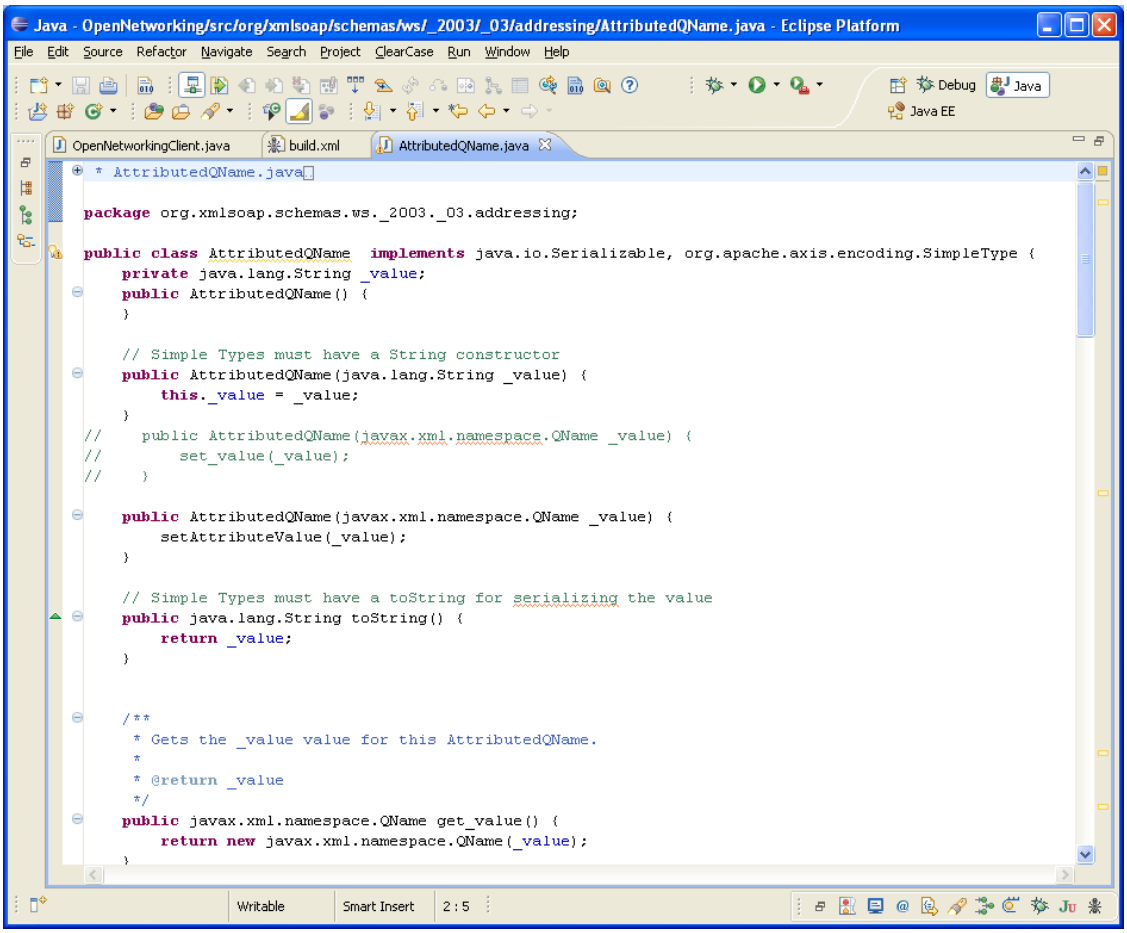


Figure 5 Generated Proxy with Error removed.

Step 3 - Create Client

7. 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 dialled 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.

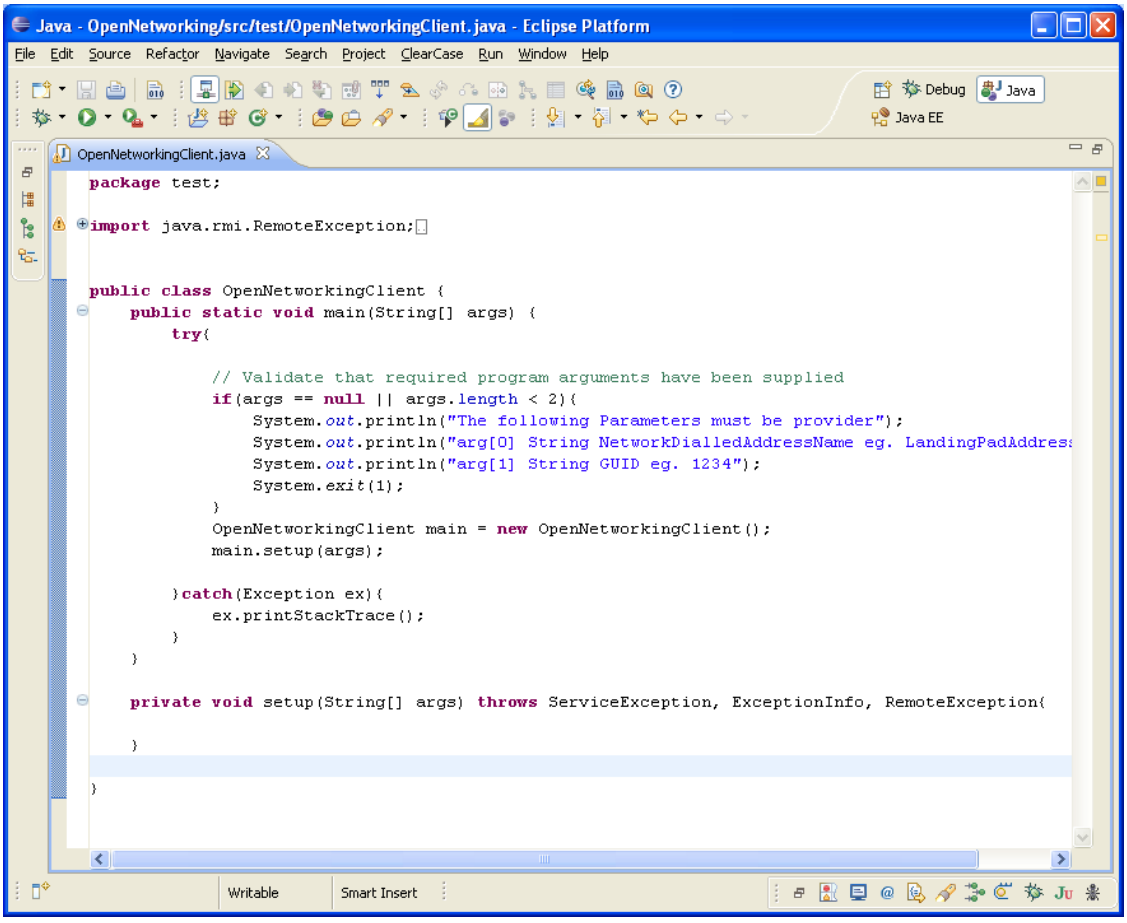


Figure 6 OpenNetworkingClient

Step 4 - Reserve Landing Pad

8. 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);
```

Step 5 – Complete Listing

9. The following is a complete listing of all the client application. The resultant project can be found at:
   /Open Interfaces Open Networking SDK/reference client

```
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();
                }
        }
}
```

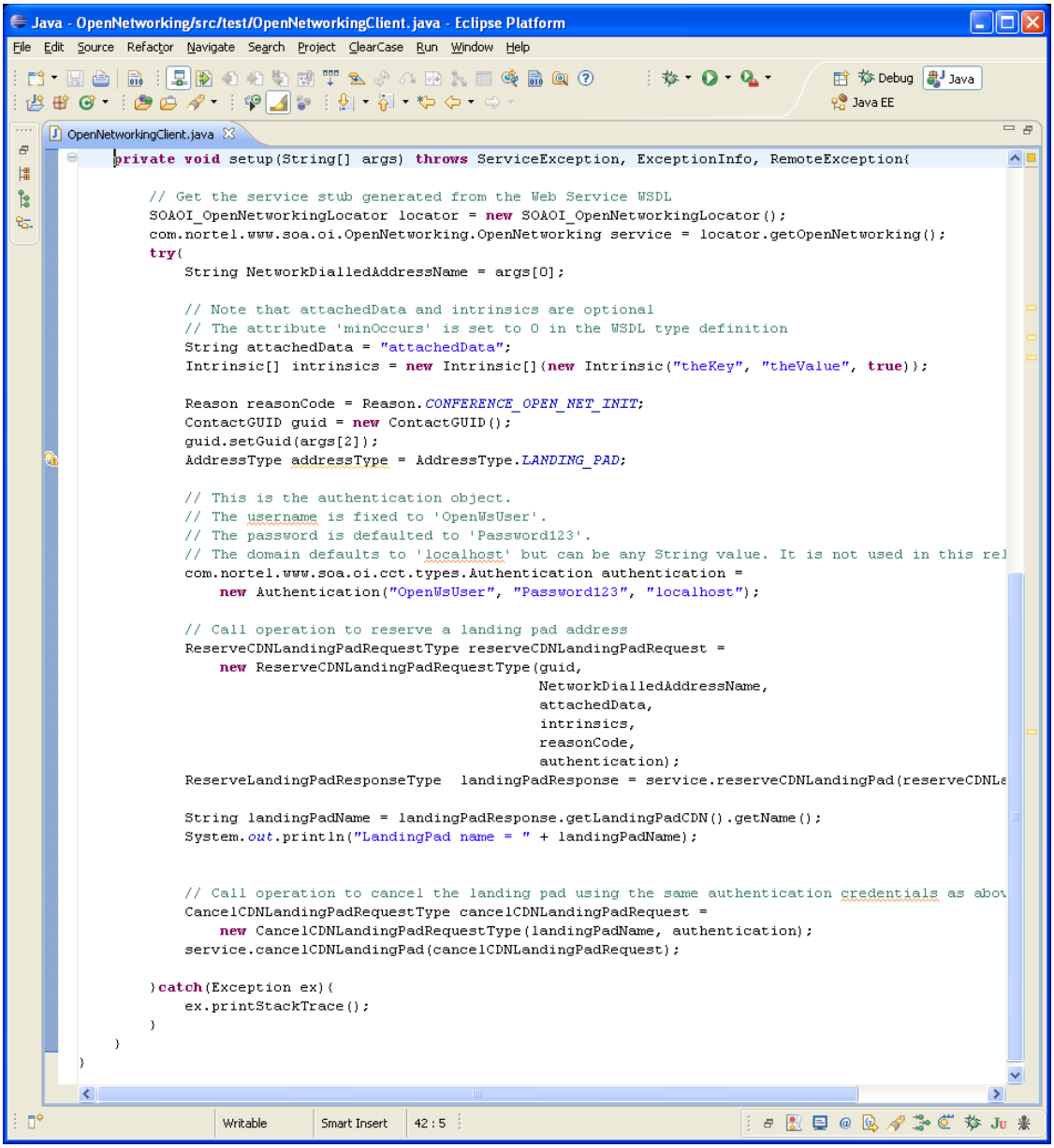


Figure 7 Code for the setup( ) Method