



Avaya Aura[®] Contact Center Open Queue Web Services

Release 7.0.0

Issue 3.0

June 2016

© 2016 Avaya Inc.

All Rights Reserved.

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](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](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.

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.

Downloading Documentation

Contents

Chapter 1: Revision History	6
Chapter 2: Overview.....	7
What is the SDK	8
SDK contents	8
SDK support	8
Related Avaya Aura Contact Center documents.....	8
Chapter 3: Open Queue Introduction	9
Some useful definitions.....	9
Intrinsics Limitations	10
Login/Logoff.....	10
CreateOQContact.....	10
GetOQContact, DropOQContact	11
Adding and Removing event listeners.....	11
Chapter 4: Reference Client	12
Programming with the Open Queue Open Interfaces	13
<i>Prerequisites</i>	13
Open Interfaces Open Queue Eclipse Project	13
Chapter 5: Tutorial	14
Assumptions about this tutorial:	14
Prerequisites	14
Tools Used	14
Using the SDK reference client sample code	15
Creating an Eclipse project for the application	16
Creating Web Service Proxy classes	17
Create the Application	18
<i>Step 1 - Create the Class</i>	18
<i>Step 2 - Create the Service</i>	19
<i>Step 3 - Authenticate User</i>	20
<i>Step 4 – Import the sample intrinsics file</i>	21
<i>Step 5 – Add the Contact</i>	23
<i>Step 6 – Logout</i>	23
<i>Step 7 – Putting it all together</i>	24
Chapter 6: Troubleshooting	27

Cannot access the Open Queue WSDL	27
Troubleshooting the “Max number of logins reached” error message	29
Troubleshooting the “InvalidArgumentException” error message	29
Troubleshooting the “User was not found” error message	29
Troubleshooting the “invalid RoutePointAddress” error message	29
Troubleshooting Open Queue scripts	30
Troubleshooting a “HTTP transport error: java.net.UnknownHostException” message	30
Troubleshooting when agents do not receive queued contacts	30

Chapter 1: Revision History

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

Chapter 2: Overview

The Open Queue Web Services allow third-party application developers the ability to queue their own contact-types in the Contact Center to be queued to available agents. Customers can queue these third-party contact types into the Contact Center to be routed to skilled idle agents using criteria specified by the application as part of the contact creation or by standard scripting.

This web service is hosted on the CCMS server and requires that the Open Queue and Open Interface Open Queue feature be enabled on the CCMS server.

This web service can be optionally configured to use TLS, if 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 SOA is configured and enabled on CCMS a list of services can be obtained through the splash screen at the following location, where the *<CCMS HostName>* is the host name of the CCMS server.

`http(s) ://<CCMS HostName>:9070`

Each service is defined by a WSDL. This WSDL (Web Service Definition Language) 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.

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/SOA0I/services/OpenQ?wsdl`

This WSDL (Web Service Definition Language) 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 Queue Web Services SDK can be downloaded to your client machine and contains information about how third-party applications can call the Open Queue Web Services.

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 Queue, 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 Queue Introduction

The Open Queue Web Services allow third-party application developers the ability to queue their own contact-types in the Contact Center to be queued to available agents. Contacts created using the Open Queue Open Interface can be reported on independently of contacts already in the Contact Center.

When a contact is created using the Open Queue Open Interface it is anchored to a specific instance (OpenQRoutePointAddress) of a Route Point Address where it remains until it is ultimately routed to an idle agent. While the contact is waiting to be routed a component called the Agent Skillset Manager will attempt to reserve an idle agent using a scripting interface exposed by the workflow component. When an agent is reserved the routing of the contact from the route point to the destination is performed by the Task Flow Executor where the Agent is ultimately notified of the new contact by a screenpop issued by the Communication Control Toolkit.

Before a third-party application will be able to queue a contact within the contact center it must first follow some steps

1. **Login/Logoff** – On successful authentication with the web service the third-party application receives a token to identify itself
2. **CreateOQContact** - issue the create open queue contact request.
3. **GetOQContact, DropContact**, - optionally query or remove existing contacts
4. Adding and Removing event listeners - optionally register interest for related events.

Some useful definitions

Term	Definition
Contact Center contacts	A contact represents a media agnostic session with an 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.

Intrinsics Limitations

It is not recommended that intrinsics should be used to store large quantities of data as it may impact overall performance in the Contact Center.

If these intrinsics are to be accessed using the scripting engine contained in Contact Center then there are additional limitations to the size and amount of intrinsics that can be associated with a contact, refer to the scripting documentation for further details.

Login/Logoff

Before the service can be used the developer must first receive a Single Sign On (SSO) token that will be used for all subsequent calls to the service. To receive this token the developer must supply the required authentication details;

- Username – **OpenWsUser** is the fixed user name that is associated with this service. Only one application at a time is allowed login using this username.
- Password – **Password123** is the default password associated with this service. This password can be changed in the *WS Open Interface* dialog of the *Contact Center Service Configuration* application located on the CCMS server.
- Domain – **open_queue** this field is used as a qualifier to differentiate this service from other services that also use the OpenWsUser username.

The Open Queue Open Interface only allows one user to be logged on at any one time, hence there can only ever be one sso token associated with an active session. If the user tries to log in when there is currently an active session they will receive an exception detailing that only one user can login at any one time.

Each session has a time out associated with it. The default for a Session timeout is 2 hours but can be configured using the *WS Open Interface* dialog of the *Contact Center Service Configuration* application on the CCMS service. Each time the service is accessed this timeout is reset.

When a user/application is exiting the application should logout the Open Queue session to ensure they do not receive an error the next time they login. If the application crashes and the user is unable to logout the session then they can use the method `logoffSession` to logoff the session without the need for the sso token.

CreateOQContact

The *CreateOQContact* will issue a request to the Open Interfaces Open Queue web service to create a contact in the Contact Center.

The *createOQContact* requires four parameters:

- `externalContactId` – a unique id for third-party applications to identify and retrieve the newly created contact using either the *getOQContact* or the *dropOQContact*.
- `outOfProviderAddressName` – the name typically is the address name of the client originating the contact into the Contact Center.
- `intrinsic` – an array of key value pairs that are associated with the contact that

- can be used to influence the route decision of the call.
- `ssoToken` – a unique identifier used to validate the current users session.

To prevent the Open Queue Open Interface adversely affecting the Contact Center traffic there is a non-configurable limit of a maximum of 20 contacts that can be created per second.

GetOQContact, DropOQContact

Using the *externalContactId* provided in the *createOQContact* a third-party application can retrieve the newly create contact or can issue a request for the contact to be removed from the Contact Center.

Adding and Removing event listeners

Third-party applications will be able to register endpoints with the Open Queue Open Interfaces. These endpoints allow applications to receive asynchronous events regarding the status of the Provider. The provider ultimately processes all incoming contact addition and removal request.

Chapter 4: Reference Client

The Open Queue Open Interfaces contains a reference client that can be used to place a sample contact with the Contact Center.

This reference client can be executed by calling the batch file [OIOQ_createContact.bat](#) with the following parameters.

Parameter	Description
External Id	A unique identifier that to associated with the Open Queue contact.
Out of Provider Address	The name of the originating address
WSDL Server IP (if not using the default WSDL location)	The IP address of the CCMS server hosting the web service.

Programming with the Open Queue Open Interfaces

The Open Queue Open Interfaces provides a powerful SOAP base web service interface that allows third-party application developers to integrate their application with Avaya Aura Contact Center Open Queue.

Prerequisites

The following assumptions are made.

1. That the Open Queue Open Interfaces are configured and running. Refer to the section Related Documents on how to configure the web services.
2. That the developers are both familiar with their choice of technology and how that technology integrates with web services.

Open Interfaces Open Queue Eclipse Project

This SDK contains a reference implementation of the Open Queue Open Interfaces. This reference implementation shows how the Open Queue web services can be called from a java client.

Chapter 5: Tutorial

This tutorial describes how to utilize the features provided by the AACC Open Queue Web Services. It will show how to generate the necessary proxies to create a simple application that will authenticate itself with the service and create a new Open Queue contact.

Once the application is launched it will login into the Open Queue web service and create an Open Queue contact for each row in the `intrinsic.csv` file.

Assumptions about this tutorial:

- The WSDL for Open Queue 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 Open Queue is configured and that the user also has a license for Open Queue web services.

Tools Used

This tutorial was created using

- Avaya Aura Contact Center 7.0
- Apache CXF
- Eclipse IDE

Using the SDK reference client sample code

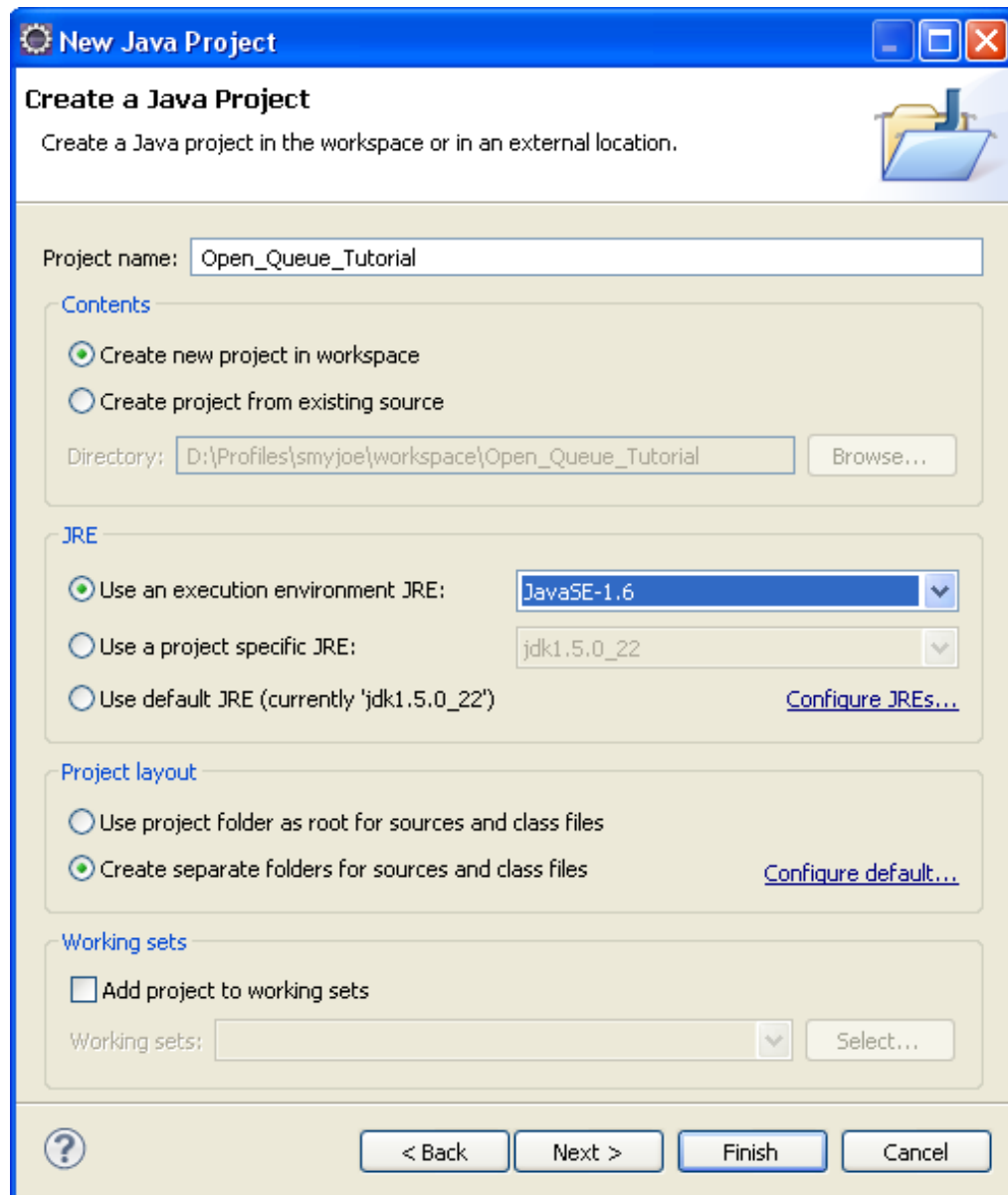
Follow these steps to use the SDK reference client sample code.

- 1) [Creating an Eclipse project for the application](#)
- 2) [Creating Web Service Proxy classes](#)
- 3) [Create the Application](#)
 - a. [Create the Class](#)
 - b. [Create the Service](#)
 - c. [Authenticate User](#)
 - d. [Import the sample intrinsics file](#)
 - e. [Add the Contact](#)
 - f. [Logout](#)
 - g. [Putting it all together](#)

Creating an Eclipse project for the application

Launch Eclipse and create a new Java Project using the default settings.

1. Open Eclipse IDE.
2. Select **File > New > Java Project**.
3. Select **Next**.
4. Enter Project Name `Open_Queue_Tutorial` and select a desired location
5. Select **Next**, **Next** and **Finished**.
6. Add a new **src** folder : Right click on the project tab, Select **New > Src Folder**. Call this new folder `autogen.src`. This is where our generated proxies will be stored.



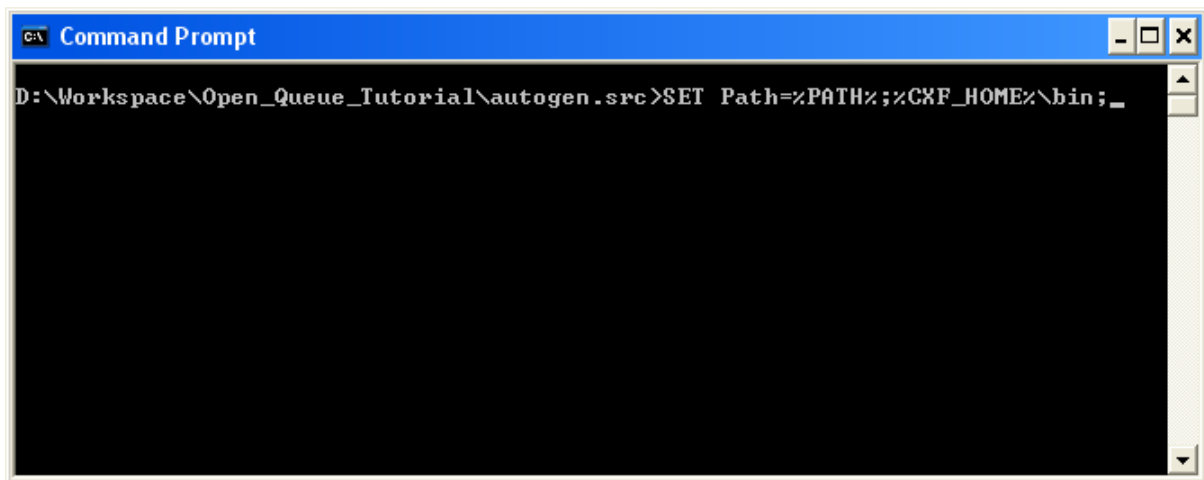
Creating Web Service Proxy classes

Applications can communicate with web services directly using SOAP messaging, for simplicity we are going to generate proxy classes to abstract us away from the underlying messaging. There are many different tools available to assist a developer in generating the client proxies but in this example we are going to use CXF.

For this step we require that CXF is installed on your system and the CXF_HOME is configured. Please refer to the CXF documentation for more details.

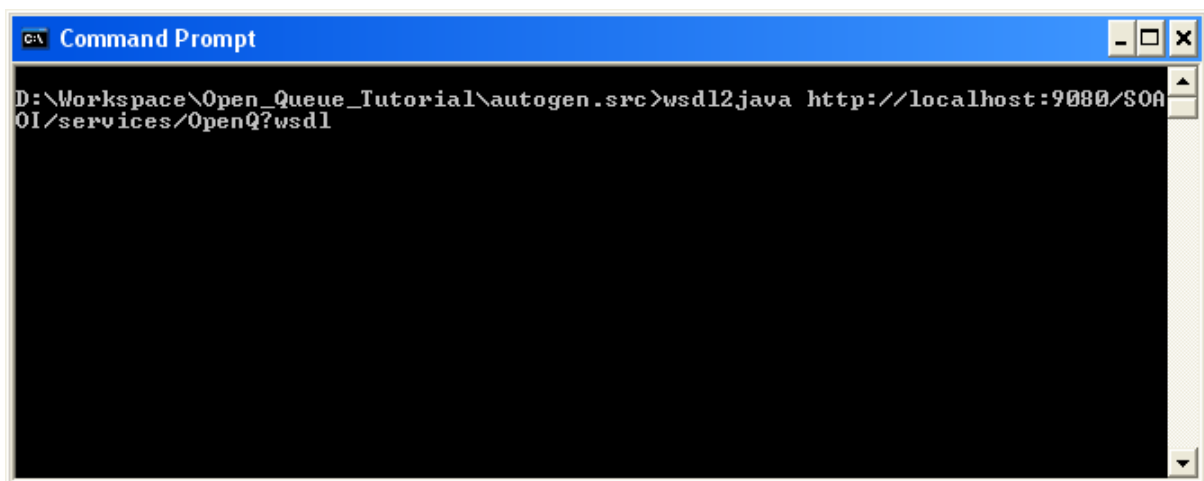
- Open a command window in the newly created autogen.src directory.
- Set the system PATH to the CXF bin directory:

```
SET Path=%PATH%;%CXF_HOME%\bin;
```

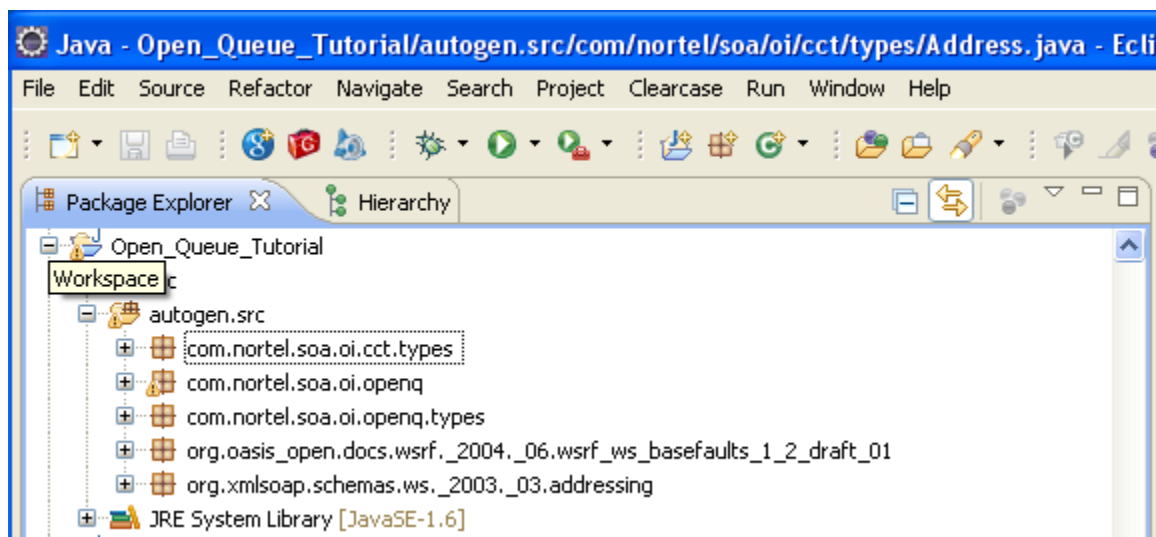


- To generate the Proxies we use a tool called wsdl2java. This tool will generate the proxies from the downloaded WSDL or by supplying the service URL. For this application we will supply the url

<http://localhost:9070/SOAOI/services/OpenQ?wsdl>



- When the tool has completed you will see two new packages with a series of java classes. After refreshing the project these packages can be seen in Eclipse.



Create the Application

Create a main class that will process a comma-separated values file to create a series of contacts using the Open Queue web service. To call the Open Queue web service you will typically need to perform the following steps.

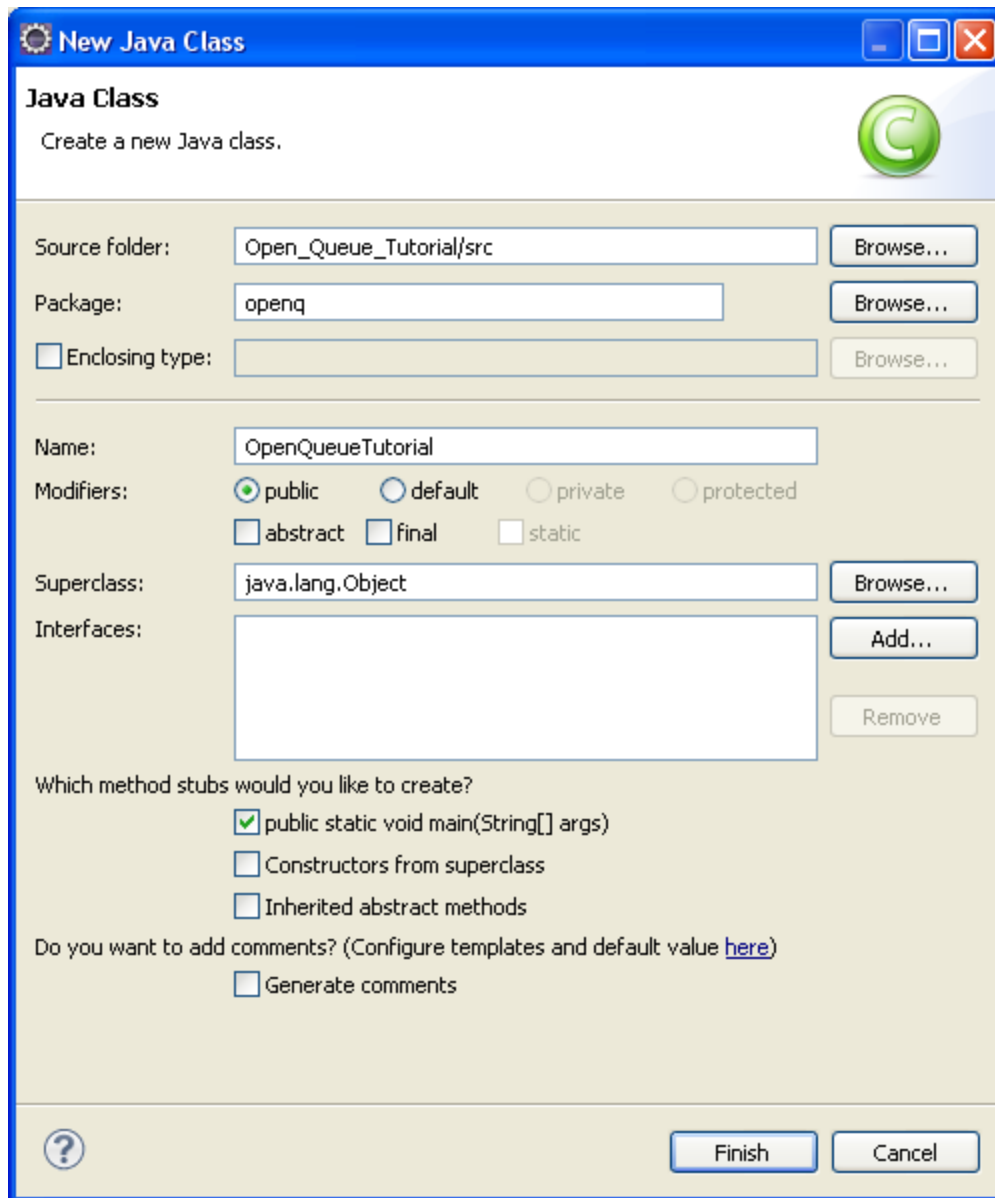
- **Create the Service** – configure the proxies to point to the CCMS server hosting the service.
- **Login** – Authenticate the application with the Web Service to retrieve an *sso (single sign on) token*.
- **Add the Contact** – call the method to add a new contact
- **Logout** – logout of the service to conserve resources.

In this project we have added an additional step to process the csv file.

Step 1 - Create the Class

In this step we are going to create a new main class and authenticate ourselves with the user to receive and an sso token

1. Create a new Java class called OpenQueueTutorial with a package openq



Step 2 - Create the Service

Before we can call the methods on the service we must first point our application at the WSDL location. To do this we must supply the hostname of the CCMS server and the port on which the service is located, default is 9070.

1. Create a global variable with the CCMS host name, web service port and a variable for the service.

```
static String host = "localhost:9070";  
static OpenQ service = null;
```

2. Create a new method called `createService`.

```
public static void createService() throws MalformedURLException,
    GetVersionFault{

    String serviceUrl = "http://" + host +
        "/SOA/IO/services/OpenQ?wsdl";

    URL url = new URL(serviceUrl);

    service = new SOAIOpenQ(url).getOpenQ();

    System.out.println("createService(): service[" +
        (service == null ? "null": service.getVersion(new
        GetVersionRequest())) + "]");

}
```

3. When this method is called successfully it will create a new service point to the configured URL

Step 3 - Authenticate User

In this step we must authenticate our application with the CCMS service to receive an sso token. This token is used in all subsequent method calls to keep track of the application session.

1. Create the following global variables. These variables contain the username and password for the application. The username is always **OpenWsUser** but the password can be changed through the CCMS Server Configuration application

```
static String username = "OpenWsUser";
static String password = "Password123";
static String domain = "open_queue";
static SsoToken sso = null;
```

2. Create a new method called login

```
public static void login() {
    try{
        AuthenticationLevel details = new AuthenticationLevel();
        details.setUsername(username);
        details.setPassword(password);
        details.setDomain(domain);
        sso = service.logIn(details);
    } catch (LogInFailedFault liex) {
        System.out.println("A login error has occurred. This may
        indicate another application is currently logged in.");
        System.out.println("Please try again in a few seconds.");
        System.out.println("If the problem persists add the value logout
        to forcibly logout the existing session i.e. SalesforceOpenQ '"
        + strFile + "' '" + host + "'logout'");
        System.out.println("error[" + liex + "]");
        System.exit(0);
    }
}
```

3. When this method is successfully called it supplies the username and password to the service and receives an sso token in return.

Step 4 – Import the sample intrinsics file

In this step we are going to open a text file called *intrinsics.csv*. This text file contains an external Id for the contact that will be created and a series of intrinsics to be associated with this contact.

An intrinsic is a key/value pair. These intrinsics offer application developers a way to associate business information with a contact. This information can be used in routing decisions or viewed by the agent when the contact is answered. Each key must have its own unique name and can have any text value associated with it.

The following sample file will create two contacts with external Ids 1 and 2. Each of these contacts will contain two intrinsics called *intrinsic_name1* and *intrinsic_name2*

```
id,intrinsic_name1, intrinsic_name2
1,intrinsic_value1, intrinsic_value2
2,intrinsic_value1, intrinsic_value2
```

1. In the main method add code to open the file and process each of the rows in the *intrinsics.csv* file.

```

BufferedReader br = new BufferedReader( new FileReader("intrinsic.csv"));

String strLine = "";
StringTokenizer st = null;
IntrinsicArray intrinsicArray = new IntrinsicArray();
List<Intrinsic> intrinsics = intrinsicArray.getItem();
int lineNumber = 0, tokenNumber = 0;
String contactId = null;
Contact contact = null;

//read comma separated file line by line
while( (strLine = br.readLine()) != null) {

    lineNumber++;
    //break comma separated line using ","
    st = new StringTokenizer(strLine, ",");

    while(st.hasMoreTokens()){
        tokenNumber++;
        intrinsics.clear();

        // Process Header
        if(lineNumber == 1){
            headerMap.put(tokenNumber, st.nextToken());
        }else{
            //display csv values

            if(tokenNumber == 1){
                contactId = st.nextToken();
            }else{
                Intrinsic intrinsic = new Intrinsic();
                intrinsic.setKey((String)headerMap.get(tokenNumber));
                intrinsic.setValue(st.nextToken());
                intrinsic.setImmutable(true);
                intrinsics.add(intrinsic);
            }
        }

    }

} // end while
//reset token number
tokenNumber = 0;
}

```

Step 5 – Add the Contact

In this step we will add a new Open Queue contact for each *external Id* present in the *intrinsic.csv* file. If a contact with a matching *external Id* exists in the Contact Center, an exception will be thrown, otherwise a new contact will be created.

1. Create a new method called *addContact*. The parameter *OutOfProviderAddressName* indicates the source of where this contact came from, similar to a calling address, this value is set to "SourceName" here.

```
public static Contact addContact(String id, IntrinsicArray intrinsics)
    throws CreateOQContactFailedFault{
    return service.createOQContact(id, "SourceName", intrinsics, sso);
}
```

2. When this method is successfully called it will return the newly created contact.

Step 6 – Logout

Only one active session is allowed to utilize the Open Queue web service so it is important to logout when the application is complete to free up resources. Applications can logout by either supplying the sso token or by supplying the login credentials. In this step we are going to logout using login credentials to allow us to forcibly remove any older sessions.

1. Create a new method *logout*.

```
public static void logout() {
    try {
        AuthenticationLevel details = new AuthenticationLevel();
        details.setUsername(username);
        details.setPassword(password);
        details.setDomain(domain);

        com.nortel.soa.oi.openq.types.LogOffSessionRequestType request = new
        com.nortel.soa.oi.openq.types.LogOffSessionRequestType();
        request.setAuthenticationLevel(details);
        // log off session
        service.logOffSession(request);
    } catch (LogOffSessionFailedFault e) {
        // TODO Auto-generated catch block
    }
}
```

```

        e.printStackTrace();
    }
}

```

2. When this method is successfully called the sso token will be invalidated. If the application fails to logout the session will timeout after a specified period, the default is 2 hours.

Step 7 – Putting it all together

Now that the methods are created, put them all together.

1. The following is the complete listing for the main method

```

public static void main(String[] args) {
    try{
        if(args == null || args.length ==0){
            System.out.println("default args[1] - host[" + host +"]");
            System.out.println("OpenQueue Web Service url[http://" + host +
                "]/SOA/Services/OpenQ?wsdl");
        }else{
            if(args.length >=1){
                host = args[1];
            }
        }

        // STEP 2 - Create the Service
        createService();

        // STEP 3 - Authenticate User
        login();

        // STEP 4 - Import the file
        HashMap headerMap = new HashMap();

        //create BufferedReader to read csv file
        BufferedReader br = new BufferedReader( new FileReader("intrinsic.csv"));
        String strLine = "";
        StringTokenizer st = null;
        IntrinsicArray intrinsicArray = new IntrinsicArray();
        List<Intrinsic> intrinsics = intrinsicArray.getItem();
        int lineNumber = 0, tokenNumber = 0;
    }
}

```



```

String contactId = null;
Contact contact = null;

//read comma separated file line by line
while( (strLine = br.readLine()) != null) {
    lineNumber++;
    //break comma separated line using ","
    st = new StringTokenizer(strLine, ",");

    while(st.hasMoreTokens()){
        tokenNumber++;
        intrinsics.clear();
        // Process Header
        if(lineNumber == 1){
            headerMap.put(tokenNumber, st.nextToken());
        }else{
            display csv values

            If(tokenNumber == 1){
                contactId = st.nextToken();
            }else{
                Intrinsic intrinsic = new Intrinsic();
                intrinsic.setKey((String)headerMap.get(tokenNumber));
                intrinsic.setValue(st.nextToken());
                intrinsic.setImmutable(true);
                intrinsics.add(intrinsic);
            }
        }
    } // end while

    // STEP 5 - Add the Contact
    try{
        // ignore the header
        if(lineNumber > 1){
            contact = addContact(contactId, intrinsicArray);
        }
    }catch(Exception ex){

```

```

System.out.println("Error: Unable to Create contact with
externalId[" + contactId +"], error[" + ex.getMessage() + "]);
}
if(contact != null){
    System.out.println("created contact id[" +
contact.getId()+ "], externalId[" +
contact.getExternalContactId() + "].");
}
//reset token number
tokenNumber = 0;

}
}
catch(Exception ex){
    ex.printStackTrace();
}finally{
    // STEP 6 - Logout
    logout();
}
}
}

```

2. On successful completion of the method, the following output should be seen on the console.

```

default args[1] - host[localhost:9070]
OpenQueue Web Service url[http://localhost:9070]/SOA_OI/services/OpenQ?wsdl
createService(): service[com.nortel.soa.oi.cct.types.GetVersionResponse@17b4703]
created contact id[4bf8f650-75ef-4fd1-891a-c9f29c2ed0fa], externalId[1].
created contact id[fe71bd4a-ad94-4691-a85d-f9ae3f0a0420], externalId[2].

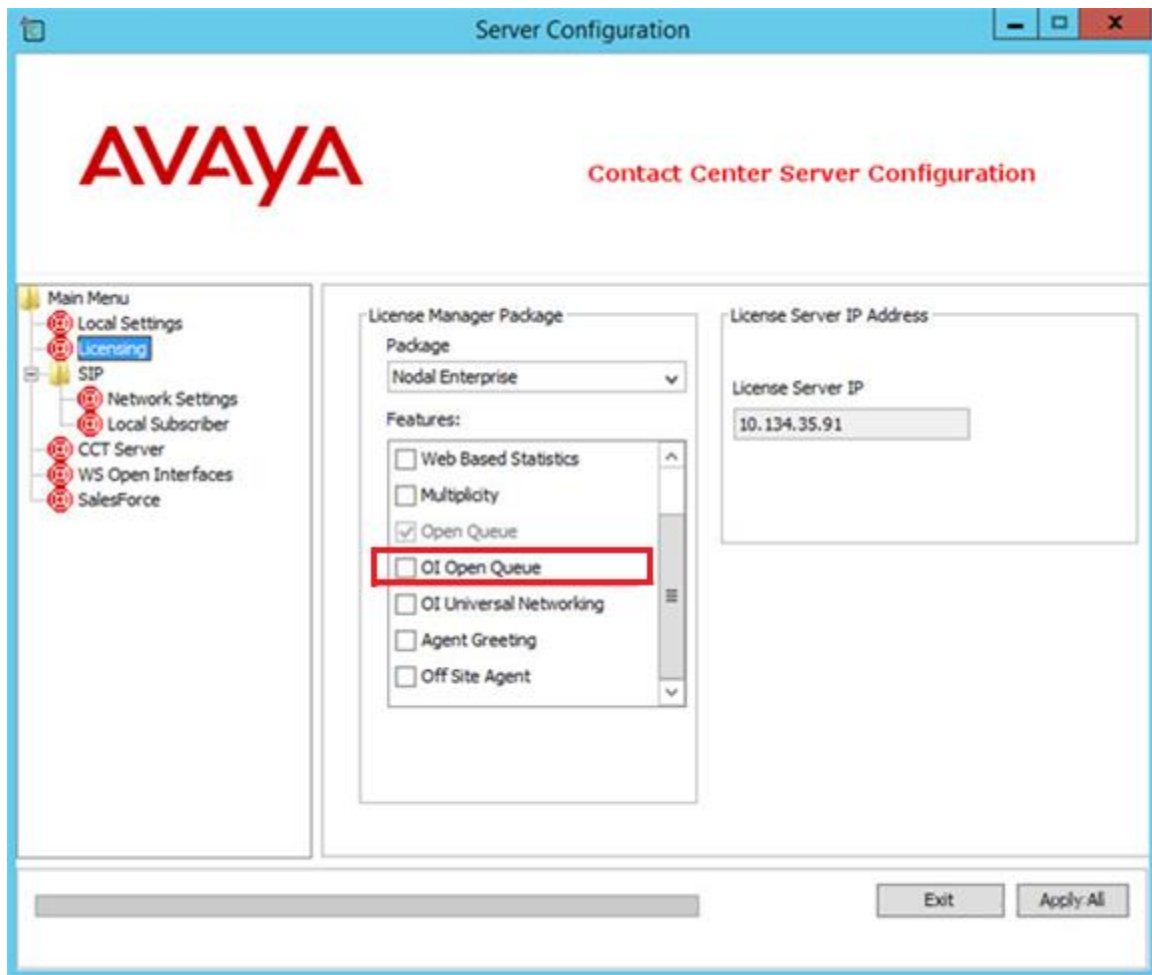
```

3. Once the contact is created the contact center will process it like any other contact and ultimately should forward it on to any agent with the OpenQ contact type.

Chapter 6: Troubleshooting

Cannot access the Open Queue WSDL

1. From the **Licensing** tab under the Server Configuration dialog on the CCMS server, verify that **OI Open Queue** 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*.

The screenshot shows the 'Server Configuration' window for 'Contact Center Server Configuration'. The 'WS Open Interfaces' tab is selected in the left-hand navigation pane. The main content area is divided into two sections: 'SOA Properties' and 'TLS Configuration [NO_CSR]'. In the 'SOA Properties' section, the 'SOA ENABLED' checkbox is checked and highlighted with a red rectangle. Other fields include 'Host' (CC7SIP), 'Ports' (9070 - 9073), 'User Name' (OpenWsUser), 'User Password' (masked with dots), 'Session Timeout' (120), and an unchecked 'TLS Encryption' checkbox. The 'TLS Configuration' section includes fields for 'Generate Certificate Signing Request' (Password and CSR File), 'Trusted Certificate Authority' (Alias and CA Cert with a 'Browse' button), and 'CSR Response Certificate' (CSR Cert with a 'Browse' button). There is also an unchecked 'Remove Certificates' checkbox. At the bottom right, there are 'Exit' and 'Apply All' buttons.

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.

Troubleshooting the “Max number of logins reached” error message

If you get the following error message “ERROR com.nortel.soa.oi.openq.OpenQImpl - logIn(AuthenticationLevel)Max number of logins reached”:

1. Only one application is permitted to access the web service for any one session. Either log out any other applications or reused the sso token that was used.

Troubleshooting the “InvalidArgumentException” error message

If you are creating an Open Queue contact and you receive the following error message “createOpenQContact():Error – InvalidArgumentException providerName[CCMM]. createOpenQContactValidate():Error - contactExternalId[10001] is already in use for an existing contact. providerName[CCMM]. “

1. A contact with a matching external Id already exists in the system. This id must be unique to the system.

Troubleshooting the “User was not found” error message

If you are creating an Open Queue contact and you receive the following error message “ERROR com.nortel.soa.oi.openq.OpenQImpl - createOQContact() :User was not found”

1. The sso token is either expired or not valid.

Troubleshooting the “invalid RoutePointAddress” error message

1. When creating OpenQ contacts, it requires that your CCMS system is configured with the following RoutePointAddress, "OpenQRoutePointAddress".
2. Log into the "Configuration" section on your CCMA web client and check within the CDN RoutePoint window that this RoutePointAddress exists. If it doesn't, add it.
Note, this should be added on new CCMS installs.

Troubleshooting Open Queue scripts

1. If you want to route open queue contacts to an agent, the agent must be associated with a skillset that supports the OpenQ contact type.
2. Use this simple script example to test your scripting and contact routing:

```
Queue to skillset Default_OpenQ  
Wait 5
```

Troubleshooting a “HTTP transport error: java.net.UnknownHostException” message

1. The WSDL used to create your application contains a reference to the CCMS host name and your network environment may not be able to resolve this host name with a supplied IP address.
2. The solution is either to configure DNS for the CCMS server or add an entry to your hosts file:

```
%SYSTEMDIR%\system32\drivers\etc\hosts
```

Troubleshooting when agents do not receive queued contacts

1. The default script that comes with Contact Center routes contacts to Agents with the OpenQ contact type configured.
2. Verify in CCMA that the agent is configured with the OpenQ contact type.
3. Verify that the script has not been change to inadvertently exclude OpenQ contact types.

LAST PAGE