SAP NetWeaver Process Integration: Using the Integration Directory API

Applies to:

EHP 1 for SAP NetWeaver Process Integration (PI) 7.1 and partly SAP NetWeaver PI 7.0, Integration Directory Application Programming Interface (Directory API).

For more information, visit the Service Bus-based Integration homepage.

Summary

This paper provides an outline to the main aspects of the SAP NetWeaver Integration Directory API for version EHP 1 for SAP NetWeaver PI 7.1 (shortly referred to as PI 7.11 in this article) and higher and provides some hints on the differences to version SAP NetWeaver PI 7.0 (shortly referred to as PI 7.0).

Author: Daniel Ritter

Company: SAP AG

Created on: 01 December 2009

Author Bio

Daniel Ritter is Software Developer at SAP NetWeaver PI.

Table of Contents

Preliminaries

Introduction / Overview

The SAP Netweaver Process Integration - Integration Directory (Builder) is the environment for configuring the communication of the Integration Server and the Advanced Adapter Engine. The set of configuration objects in the Integration Directory includes e.g. communication components (e.g. Business Systems from SLD) for senders and receivers, logical routing (receiver determination, interface determination) as well as technical routing (receiver agreement).

In addition to the Integration Builder, there is also a web service-based application programming interface (API), which can be used to modify and activate objects in the Integration Directory. The API provides the same configuration options that are possible in the Integration Directory and is implemented as synchronous web services on the java server. Thus the web service WSDLs are available in the ES Repository (or Integration Repository in the earlier version SAP NetWeaver PI 7.0) and from the java server's web service runtime.

In this document, the main focus will be on how to use the directory web service API in the PI 7.11 (and later releases) environment, giving hints how particular parts of the implementation would look like with PI 7.0. This new version of the Directory API is available with EHP 1 for SAP NetWeaver PI 7.1. When explaining the API, the corresponding parts of the Integration Directory and possibly the ES Repository (respectively Integration Repository) are set into context. However, a full comparison of the PI 7.0 and PI 7.11 (and later releases) API and explanation of the PI tools is not in scope in this document. For a comprehensive overview of the Integration Directory and API please refer to [1], [2]. When looking for a sample exclusively dealing with Directory API 7.0 see [3].

Note: In general, when using SAP Netweaver PI 7.1 (and later releases), but working with the Directory API, version 7.0, only those objects can be used, which do not use new features introduced later. For instance, trying to access attributes of sender agreements, like software component version of sender interface with schema validation, will lead to a version incompatibility exception.

The first part of the document (see Getting Started - Development Process) deals with the setup of the web service client. This is the basis for the query and read implementation (see Query and Read Configuration Content) as well as the change list administration and creation of configuration objects (see Change Lists and Write PI Configuration Content) which is explained at a sample.

Prerequisite of using the Directory API

As a prerequisite of using the Directory API, the user has to be assigned to specific roles in the java server. Hence, these roles cannot be assigned in the AS ABAP user management e.g. transaction `SU01`.

Assigning roles in PI 7.11:

Since the Visual Administrator is no longer available with PI 7.11, the SAP NetWeaver Administrator (referred to as *NWA* in this article) has to be used [3].

1. Start NWA (using http://<*server*>:<*port*>/useradmin).
2. Search for the userid.
3. Select and display the details of the userid.
4. Modify the userid.
5. In Assigned Roles tab, enter search using criteria: **SAP_XI_API***.
6. Select and add roles: **SAP_XI_API_DISPLAY_J2EE** and **SAP_XI_API_DEVELOP_J2EE**.

For the assignment of the PI 7.0 roles **(**api_develop, api_display in sap.com/com.sap.xi.directory*aii_ibdir_sbeans.jar) read [3].

Getting Started - Development Process

In this document, the Development Environment (IDE) of choice is the NetWeaver Developer Studio (NWDS) 7.1 or better 7.2. There a new project is created or an existing one is chosen into which the web service clients of the Directory API are generated. As shown in Figure 1, in the NWDS import wizard the WSDL file has to be chosen and one of the following WSDL sources is specified:

- Enterprise Services Repository (via URL)
- Remote Location / File System (WSDL file extracted from External Definition in the ES Repository: binding configuration will be required)
- Services Registry

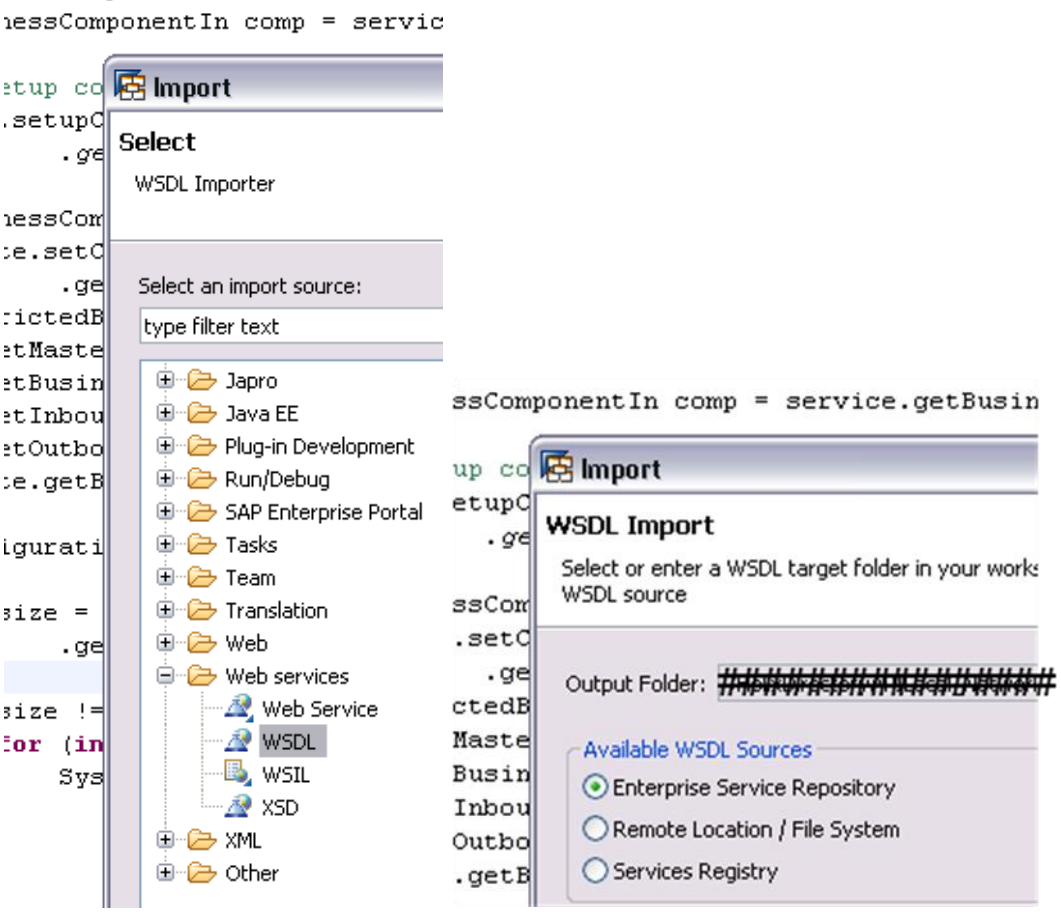

Figure 1: NWDS import wizard showing the selection of WSDL (under Web services), and the import dialog with the list of available WSDL sources

Importing WSDL from Enterprise Services Repository

The import of the WSDL directly from the Enterprise Services Repository (ES Repository) is considered as the most convenient, since all information is provided, especially the binding information, which contains the endpoint of where the web service is to be executed.

Therefore ES Repository system, host and port information has to be configured which can be done in the Enterprise Service Browser in advance. As depicted in Figure 2, this browser is located in the IDE's Preferences **»** WebServices.

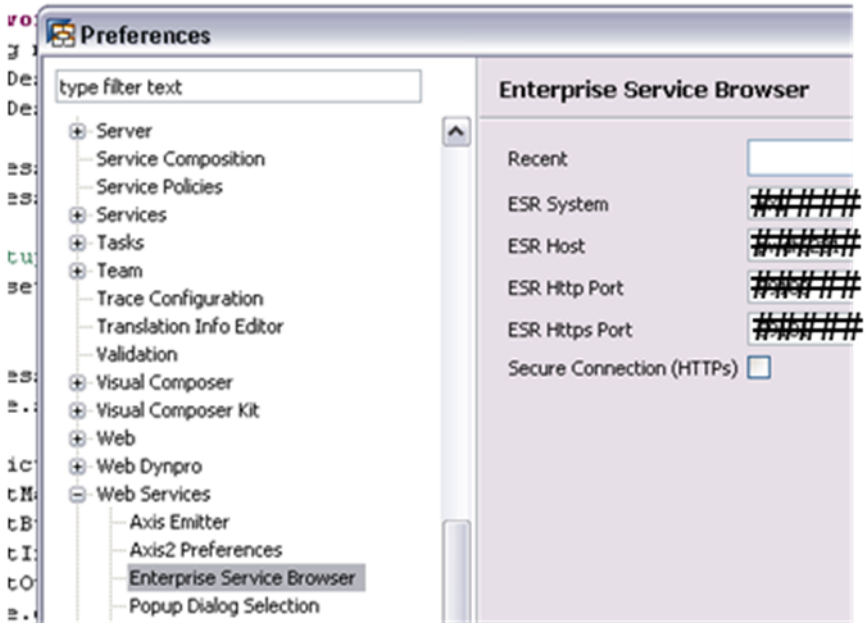

Figure 2: NWDS preferences view showing the configuration of the Enterprise Service Browser

After logging into the system's Enterprise Services Repository via NWDS import wizard, it is possible to browse through all services offered (see Figure 3) by Software Component Version and choose the service for which a client has to be generated.

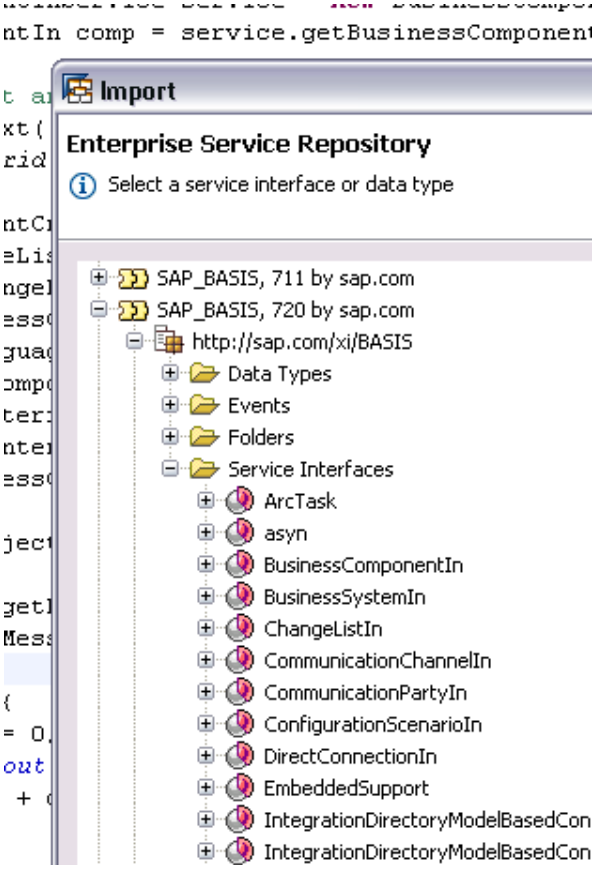

Figure 3: Import wizard using login to ES Repository showing all services available in the repository grouped by software component version

Importing WSDL from Remote Location / File System

In case of importing a WSDL from remote location / file system it is preferred to use the file from the Single Service Administration: Service Definition (SAP NetWeaver Administrator **»** Configuration Management **»** Connectivity).

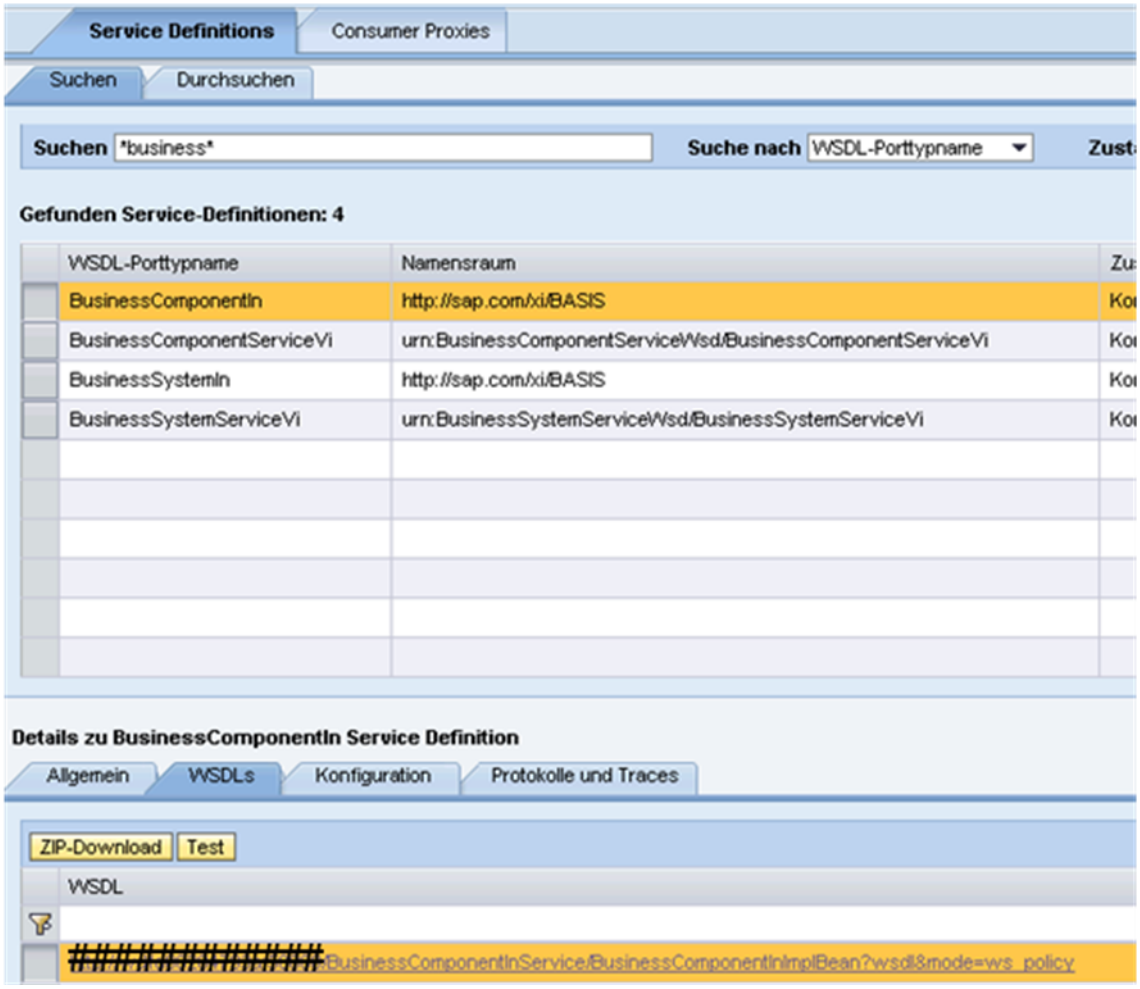

Figure 4: Service definition view of the Single Service Administration in SAP NetWeaver Administrator showing old (SAP NetWeaver 7.0) and new (SAP NetWeaver 7.11) services available in the corresponding ES Repository with their qualified WSDL. Suffix -Vi indicates old (NW 7.0) API services and suffix -In denotes new (>= NW 7.11) services

It can be loaded from the service definition using the WSDLs tab in the details section of a selected service (see Figure 4). Within this file binding information is already included, to avoid additional configuration.

After a successful import of the WSDL from arbitrary location, the web service client can be created by executing the

client generation on the file (see Figure 5, right-click on the file **»** Web Services **»** Generate Client).

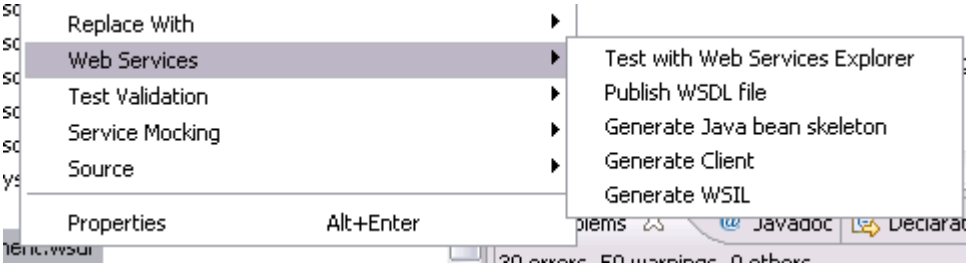

Figure 5: NWDS web service client generation possibility when right-clicking on the imported WSDL

The generation wizard (see Figure 6) is the same that is used for direct web service client creation (right-click on Project **»** New **»** Other **»** WebServices **»** Web Service Client and paste the WSDL URL from the remote location section).

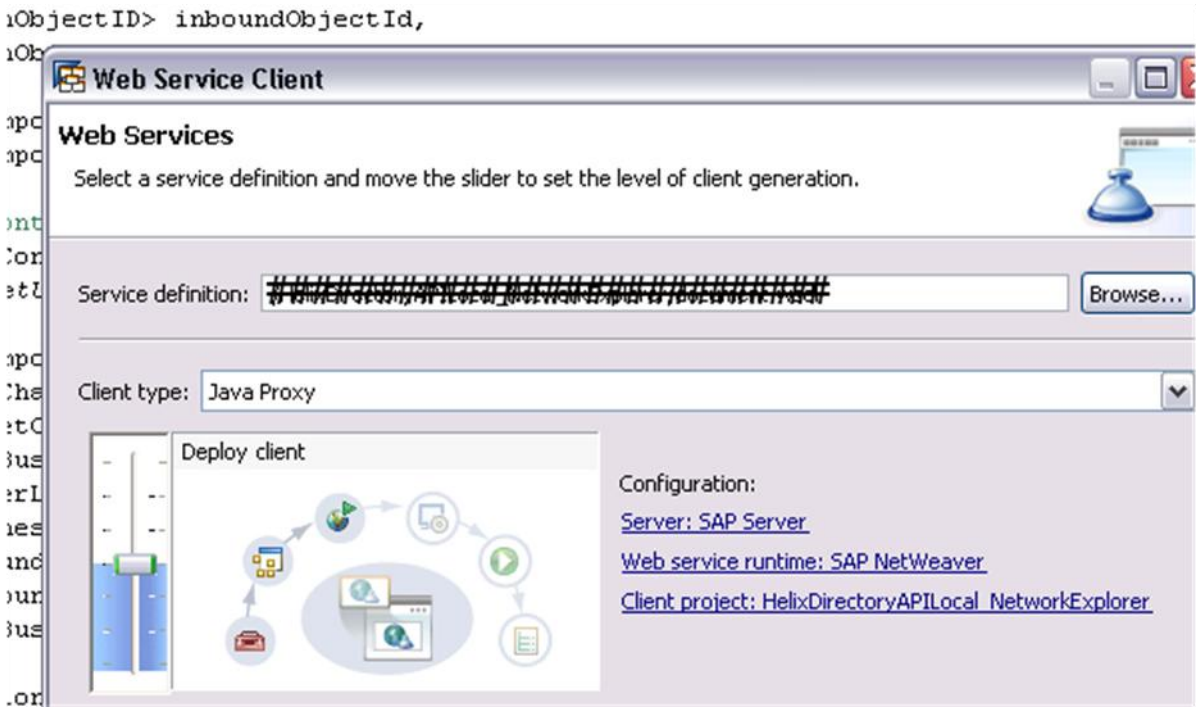

Figure 6: NWDS client generation wizard which is accessible via WSDL or directly via NWDS project showing the configuration of the client to be created

Note: if no additional web service runtime is installed on the system, the SAP NetWeaver runtime can be used. However, it is not set as default in the wizard and has to be set explicitly.

The generated web service client can now be used in any application within this NWDS project to call the Integration Directory services in the specified SAP NetWeaver system. During the development it is very helpful to use the Web Service Navigator for viewing the services and determining their obligatory parameters as well as their keys.

Query and Read Configuration Content

Since the Directory API supports all kinds of SAP NetWeaver PI configuration (configuration objects for classic PI configuration, integrated configuration for Advanced Adapter Engine (since NW 7.3), and direct connections), there are multiple entry points to access configuration data.

Although this document focuses on standard PI configuration, the gained knowledge and parts of the source code can be re-used when working with e.g. direct connections or integrated configuration. Working with the standard PI configuration objects can be done starting from configuration scenarios or straight forward by using the specific objects directly.

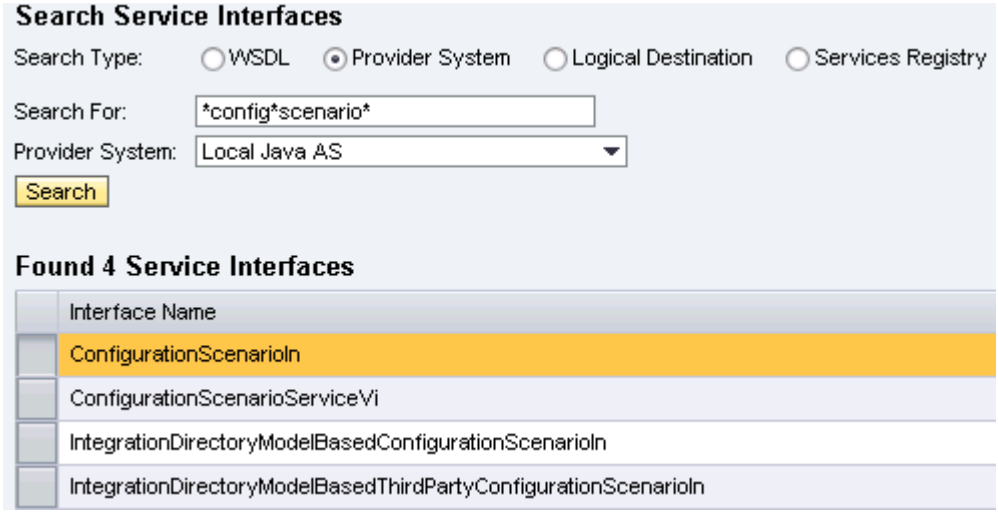

Figure 7: Web Service Navigator view accessed by Provider System on local Java AS showing the search result for configuration scenarios. Suffix -Vi indicates old (NW 7.0) API services and suffix -In denotes new (>= NW 7.11) services

When starting with configuration scenarios, using "ConifgurationScenarioIn" web service (see Figure 7), all configuration objects added to the scenario can be accessed. However, only key information is returned, which requires additional direct calls to the single configuration objects. Hence this document shows how to access the configurable objects within the standard PI configuration directly. More specifically sample access to PI communication components (business components, business systems) and the technical routing (sender/receiver agreements, sender/receiver channels) are described.

As described before, a new NWDS Java project is created called "PIDirectory_Article" with a src-package "com.sap.pi.directory.article" containing the following classes:

- DirectoryAPIMain: Main()
- DirectoryApiDAO: data access object which encapsulates the complete access to the Integration Directory
- DirectoryApiConfiguration: Parameter settings for Integration Directory access

Then the required web service clients for communication components (BusinessSystemIn, BusinessComponentIn) and technical routing (SenderAgreementIn, ReceiverAgreementIn, CommunicationChannelIn) are created from their WSDLs as shown before. In Figure 8 the NWDS project structure is denoted showing the manually created Java classes as well as the generated web service clients.

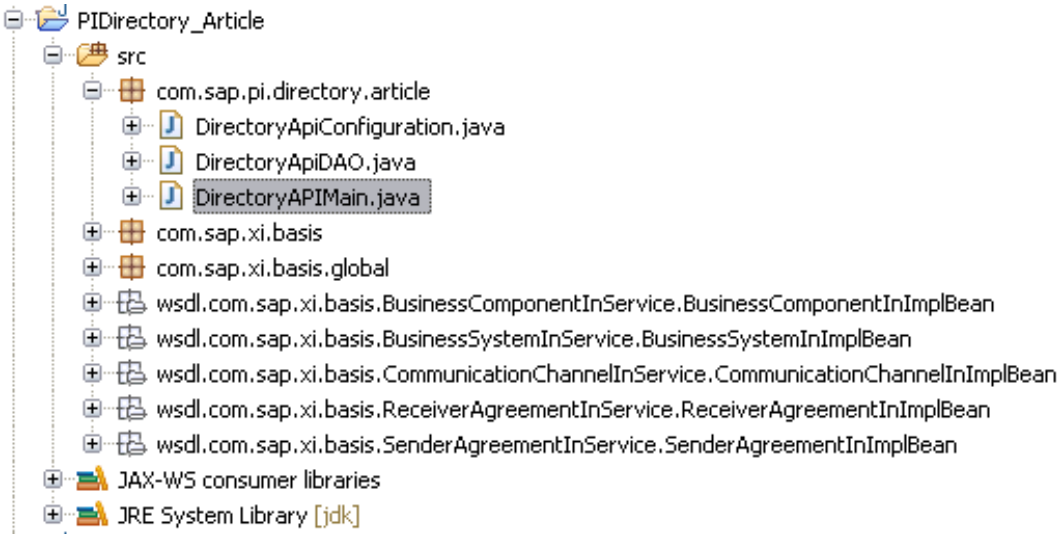

Figure 8: NWDS project structure showing manually created Java classes as well as generated web service clients

Web Service Client Configuration

Although the Integration Directory web service clients have been generated using WSDL documents from a specific ES Repository, they can be configured to access any compatible Integration Directory. Out of simplicity reasons a distinct class (`DirectoryApiConfiguration`) is created which statically manages the configuration parameters: `USERID`, `PASSWORD`, `SERVER_NAME` and `SERVER_PORT`. Of course, in productive environments this would be managed differently.

Code snippet: static member variables of the configuration class

```
private static String USERID = "youruser";
private static String PASSWORD = "yourpassword";
private static String SERVER_NAME = "yourservername";
private static String SERVER_PORT = "yourserverport";
```

In this context (see Appendix, Listing 1), `USERID` and `PASSWORD` are the same that are used to access Integration Directory. `SERVER_NAME` and `SERVER_PORT` specify the PI server and its HTTP port.

Integration Directory Data Access Object

With the configuration information, the web service clients can be used to connect to any compatible Integration Directory for offered operations {`Change`, `Check`, `Create`, `Delete`, `OpenForEdit`, `Query`, `Read`, `Revert`}. A sample implementation of the {`Change`, `Create`, `Query`, `Read`} operations for the specified configuration objects can be found in Appendix, Listing 2.

A prerequisite for all operations is the creation of a service instance

Code snippet: business system service client

```
// get web service
BusinessSystemInService service = new BusinessSystemInService();
```

which is used to retrieve a port object of this service

Code snippet: business system service client port

```
// get port
BusinessSystemIn systemComp = service.getBusinessSystemInPort();
```

and set the binding context specified in the configuration information

Code snippet: call method to setup the context of the client port

```
// setup context and authorization
this.setupContext((javax.xml.ws.BindingProvider) systemComp,
                        DirectoryApiConfiguration.getUserid(),
                        DirectoryApiConfiguration.getPassword());
```

The "setupContext" method is treating the port object as "BindingProvider", determines the endpoint address property and sets USERID, PASSWORD and the ENDPOINT_ADDRESS to it. This is basically the same mechanism as in the SAP NetWeaver PI 7.0 Directory API. The only difference apart from the different web service clients and some types is the instantiation of the port object which is done as follows:

Code snippet: Integration Directory API 7.0 sample for retrieving the port

```
// get port
BusinessSystemInServiceVi systemComp = service.getHTTPBasicAuthPort();
```

Then a request object, e.g. for query, is created filled with relevant data (if necessary) and used as import parameter when calling the service using the port object.

Code snippet: build query object and execute operation on port

```
// Create a new instance of request object expected by Communication
BusinessSystemQueryIn query = new BusinessSystemQueryIn();

// action: use systemComp to query
BusinessSystemQueryOut result = systemComp.query(query);
```

In case of a query, the result contains the keys of the found PI configuration objects as `java.util.List` of `CommunicationComponentID` and a list of log messages in case an exception occurred during execution.

Note: The log message type is `LogMessageCollection` which contains `java.util.List` for all provided PI configuration objects. However, when querying e.g. for business systems only `getLogMessageBusinessSystem()` could contain exceptions which resulted from the query.

The keys of the configuration objects can then be used to read the PI configuration object data. Hence they are added via `addAll(ids)` to the request object before execution.

Code snippet: create read object and add relevant information before executing

```
// Create a new instance of request object expected by Communication
BusinessSystemReadIn readIn = new BusinessSystemReadIn();
readIn.setReadContext(ReadContextCode.ACTIVE);
readIn.getBusinessSystemID().addAll(ids);
```

Code snippet: create read object and add relevant information before executing

An important parameter is `ReadContextCode` which specifies the change list settings {`ACTIVE`,`USER`} for user specific or active change lists.

Query and Read Integration Directory

The data access object can then be used by applications to work with the data from Integration Directory. As mentioned, the result of the query method could be used directly to retrieve data from the directory. Now, for instance, the business system identifier and its party could be printed to see all existing business systems known within the chosen Integration Directory.

Code snippet: create data access object of the Directory API and read all business systems which are returned by the query as well as print out their identifiers and parties

```
DirectoryApiDAO directory = new DirectoryApiDAO();

List<BusinessSystem> bs = directory.readBusinessSystemIn(directory
                  .queryBusinessSystemIn());

System.out.println("Business Systems:");

for (int i = 0; i < bs.size(); i++) {
      // display party and id
      System.out.println(bs.get(i).getBusinessSystemID().getPartyID()
                        + "#" +
                  bs.get(i).getBusinessSystemID().getComponentID());
}
```

For the complete sample application code see Appendix, Listing 3.

Note: When reviewing the read results of the communication channels there might be some fields which seem to lack the information which is available in the directory. It might be no surprise that passwords are not returned by API, but what about the adapter metadata?

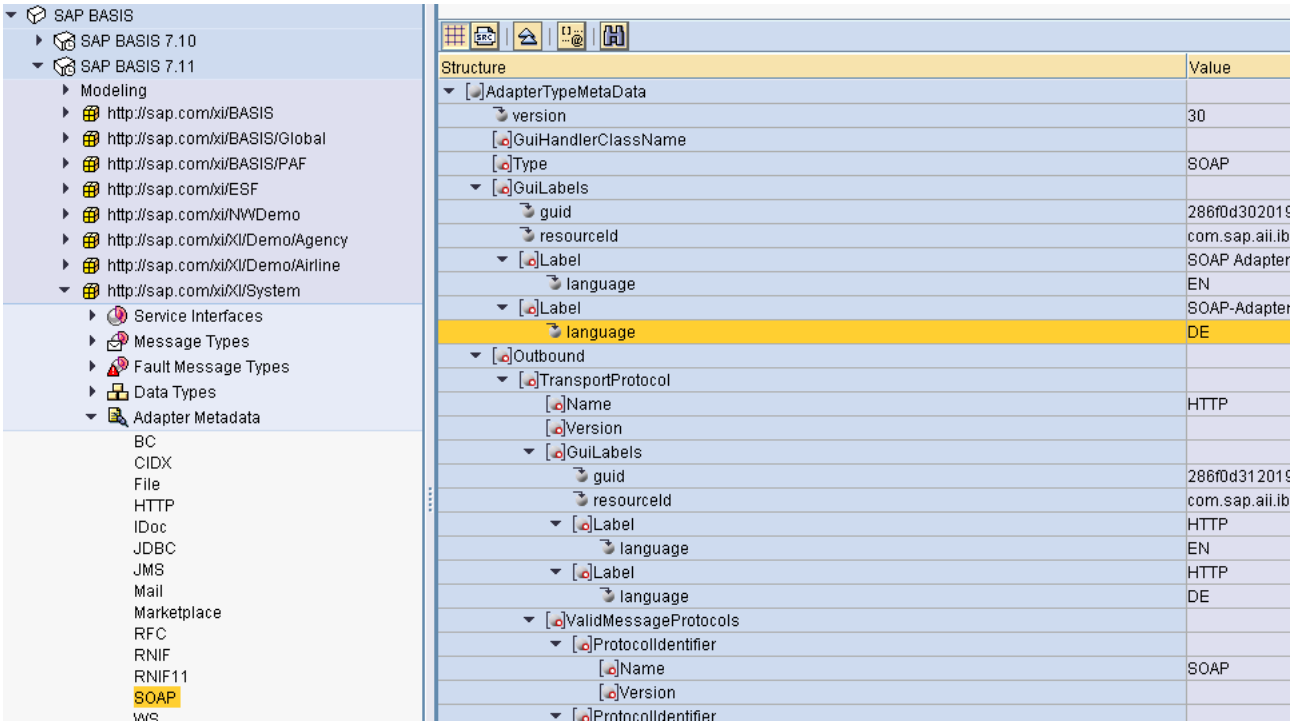

Figure 9: ES Repository view on the adapter metadata of SAP BASIS 7.11

In those cases a look into the adapter metadata view of ES Repository (see Figure 9) might be helpful, to check which information is going to be returned by the API.

Change Lists and Write PI Configuration Content

The update or creation of PI configuration content requires the use of Integration Directory change lists. Change lists are the container in which changes in general are kept until they get activated or rejected. Any of the change operations of the API create a default change list per call in which the changes are administrated. Though this default behavior might be helpful when writing sample applications, it becomes more important to control the change lists when it comes to real applications.

Using Change Lists

For the administration of change lists, a new ChangeListIn web service client is needed (see Figure 10).

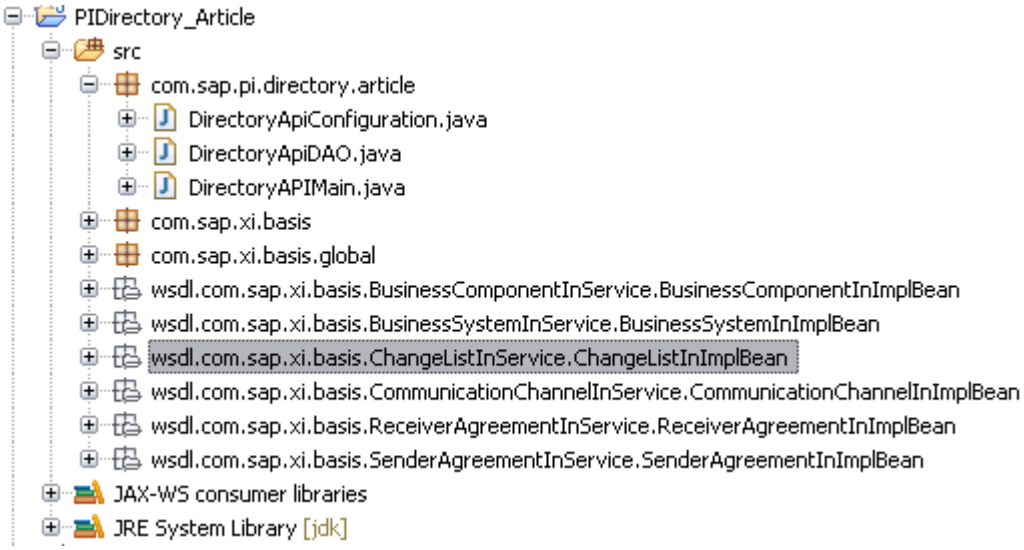

Figure 10: NWDS project structure showing the newly added web service client for change list support

The scope which is of interest for this document contains the change list's basic lifecycle operations {create, activate, reject}. As in case of the query and read operations, a service (here `ChangeListInService`) is used to get the port object. After the context has been set to the port object, a change list data transfer object of type `RestrictedChangeListID` is created which is used to specify the change list's name and language dependent description.

Code snippet: set name and language dependent description to change list object

```
RestrictedChangeListID id = new RestrictedChangeListID();
id.setName(changeListName);

LONGDescription description = new LONGDescription();
description.setValue(changeListDescription);

description.setLanguageCode(changeListDescriptionLanguageCode);
id.setDescription(description);
```

The activation and rejection of the change list then only need the ID (GUID) of the change list, which is returned when creating the list. These change list identifiers are important when it comes to the change or creation of configuration objects. For further details of the implementation see Appendix, Listing 2.

Note: The method `checkContent` can be used to check consistency of the change list and `getCacheState` returns information about the cache distribution and can thus be useful in case of erroneous behavior.

Create PI Configuration Content

After having added the change list web service client to the project, the data access object can be enhanced by the create methods for the configuration objects (see Appendix, Listing 2). The setup of service, port and context are the same as for the query and read services. But then a newly created change list is set to the request object as well as (at least) the key information required for this configuration object.

Code snippet: instantiate e.g. business system create object, set the created change list identifier, instantiate the corresponding business system data object and set its name and master language before creation

```
BusinessSystemCreateChangeIn create = new BusinessSystemCreateChangeIn();
create.setChangeListID(changeList.getChangeListID()
                        .getChangeListID());

RestrictedBusinessSystem bs = new RestrictedBusinessSystem();
bs.setMasterLanguage(masterLanguage);
bs.setBusinessSystemID(id);

create.getBusinessSystem().add(bs);

ConfigurationObjectModifyOut out = comp.create(create);
```

The same mechanisms apply to the creation of the other configuration objects. The data access object containing the complete implementation of query, read and create as well as the change list methods can be found in Appendix, Listing 2.

Case Study: Create new Receiver with XI Adapter

With the knowledge and source code that has been presented, it is now possible to create a new receiver communication component, its own receiver communication channel and a receiver agreement.
Since this case study does not assume that business systems are correctly defined in SLD, a business component will be created. A prerequisite is an existing change list `article-changeList`. See Appendix, Listing 3, for the complete source code.

Code snippet: create change list

```
// create change list
ChangeListCreateOut changeList = directory.createChangeList(
                  "article-changeList",
                  "Change list for article on Integration Directory API",
                  masterLanguage);
```

Then the business component `Receiver_Article` is created (which does not be maintained in SLD). The major difference between the creation of business systems and components is the possibility to set inbound and outbound interfaces to the business components. In this case, the business component is created with empty interfaces.

Code snippet: create change list

```
String businessComponent = "Receiver_Article";

// create new receiver business component - if it does not exist
CommunicationComponentID id = new CommunicationComponentID();
id.setComponentID(businessComponent); // does not need to exist in SLD

List<DesignObjectID> inboundObjectId = new ArrayList<DesignObjectID>();
List<DesignObjectID> outboundObjectId = new ArrayList<DesignObjectID>();

directory.createBusinessComponent(changeList, masterLanguage, id,
                  inboundObjectId, outboundObjectId);
```

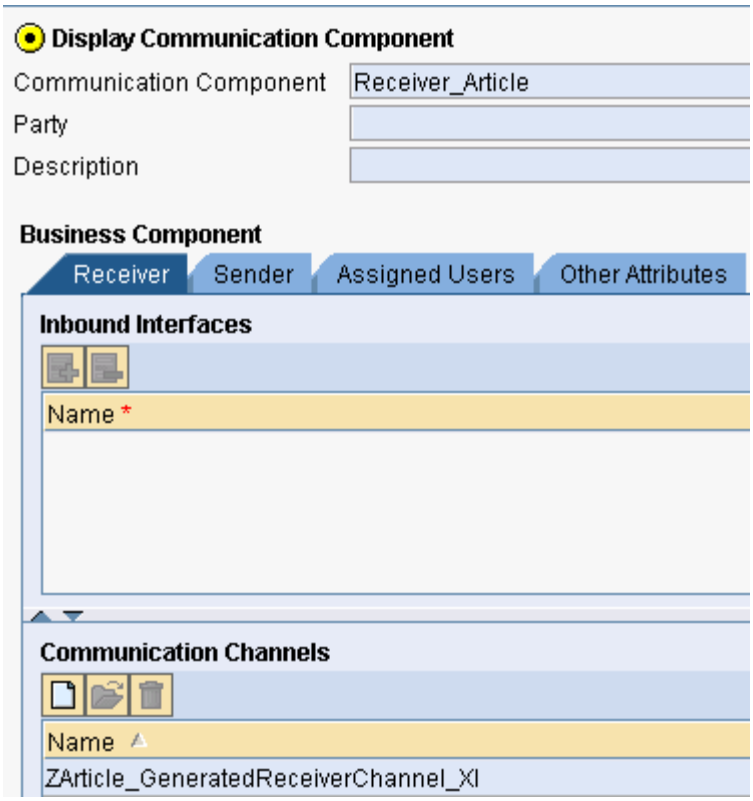

Figure 11: Communication component editor showing details of the generated business component (with its assigned communication channel). The reference to the channel is added by the system after its creation

The newly created business component (Figure 11) now needs a communication channel which we assume to be a receiver channel of type XI. Therefore only the required information is set. However, some of the adapter specific attributes, e.g. `host`, have to be set specifically for your landscape. This is required for runtime. The settings chosen in this document allow at least a creation of the communication channel.

Code snippet: create receiver communication channel of type XI

```
String receiverChannelIdString = "ZArticle_GeneratedReceiverChannel_XI";
```

```
// create RECEIVER communication channel for business system
CommunicationChannelID receiverChannelID = new CommunicationChannelID();
//receiverChannelID.setPartyID("");
receiverChannelID.setComponentID(businessComponent);
// prerequisite: is created
receiverChannelID.setChannelID(receiverChannelIdString);

// adapter meta data
DesignObjectID objectId = new DesignObjectID();
objectId.setName("XI");
objectId.setNamespace("http://sap.com/xi/XI/System");
objectId.setSoftwareComponentVersionID("e5a519a0-7661-11dc-a2e8-
e95d0a115642");

// channel data
CommunicationChannelDirection direction =
CommunicationChannelDirection.RECEIVER;

String transportProtocol = "HTTP";
String transportProtocolVersion = "1.0";
String messageProtocol = "XI";
String messageProtocolVersion = "3.0";

// adapter specific attributes
List<GenericProperty> properties = new ArrayList<GenericProperty>();

GenericProperty property = new GenericProperty();
property.setName("host");
property.setValue("TODO: set your host name");
properties.add(property);

property = new GenericProperty();
property.setName("addressingMode");
property.setValue("url");
properties.add(property);

property = new GenericProperty();
property.setName("port");
property.setValue("TODO: set your port");
properties.add(property);

property = new GenericProperty();
property.setName("path");
property.setValue("/sap/xi/engine?type=entry");
properties.add(property);

String receiverChannel = directory.createCommunicationChannel(
              changeList, masterLanguage, receiverChannelID, objectId,
              direction, transportProtocol, transportProtocolVersion,
              messageProtocol, messageProtocolVersion, properties);
```

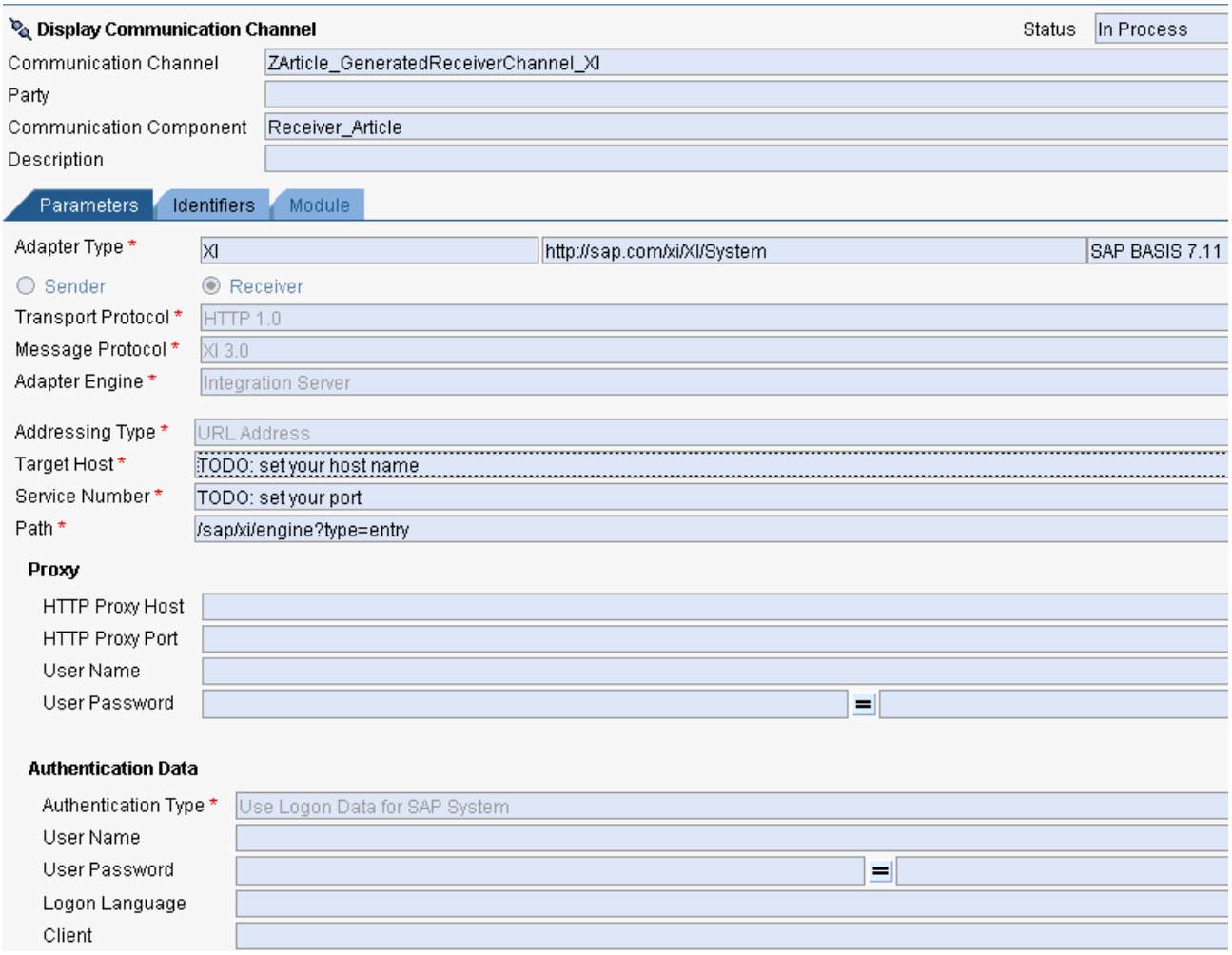

Figure 12: Communication channel editor showing details of the generated communication channel with its adapter specific attributes

With the generated receiver communication channel (see Figure 12) a receiver agreement is created assuming the receiver interfaces simply is `FlightBookingOrderRequest_In` of the delivered demo airline scenario (see Figure 13).

Code snippet: create receiver agreement for business component with communication channel

```
// create receiver agreement
// receiver agreement id
MessageHeaderID header = new MessageHeaderID();
header.setSenderComponentID("*");
header.setInterfaceName("FlightBookingOrderRequest_In");
header.setInterfaceNamespace("http://sap.com/xi/XI/Demo/Airline");
header.setReceiverComponentID(businessComponent);

// communication channel
CommunicationChannelID channelId = new CommunicationChannelID();
//channelId.setPartyID("");
channelId.setChannelID(receiverChannel);
channelId.setComponentID(businessComponent);

directory.createReceiverAgreement(changeList, masterLanguage, header,
                  channelId);
```

```
// activate changes if required
//directory.activateChangeList(changeList);
```

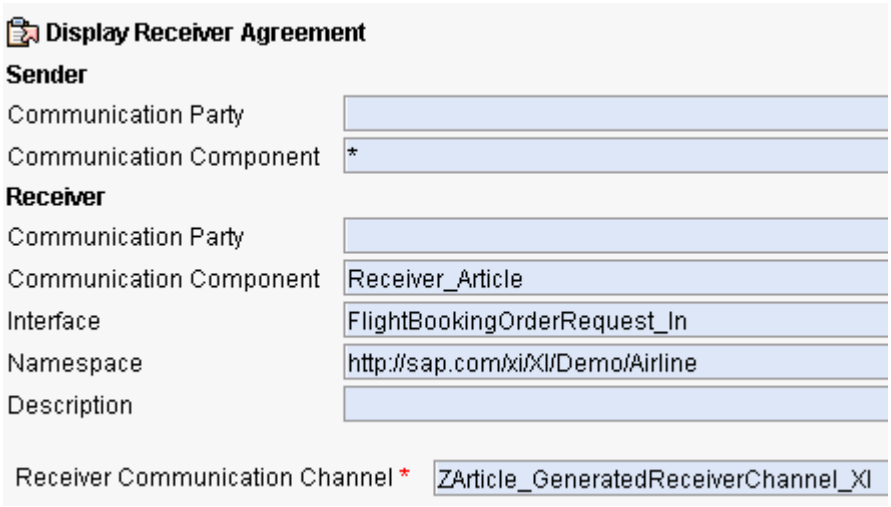

Figure 13: Receiver agreement editor showing details of the generated receiver agreement with its receiver interface and receiver communication channel

After the business component, its receiver channel and agreement have been successfully created, it is kept in the formerly created change list `article-changeList` (see Figure 14). All configuration objects can now be checked for consistency. That is due to the commented activation step in the source code and allows a manual check in the system (can be checked via `checkContent` method of the Directory API). The check only looks for severe issues and allows e.g. incorrectly filled adapter specific attributes in the communication channel. However, if the settings are not corrected, this will lead to errors during runtime.

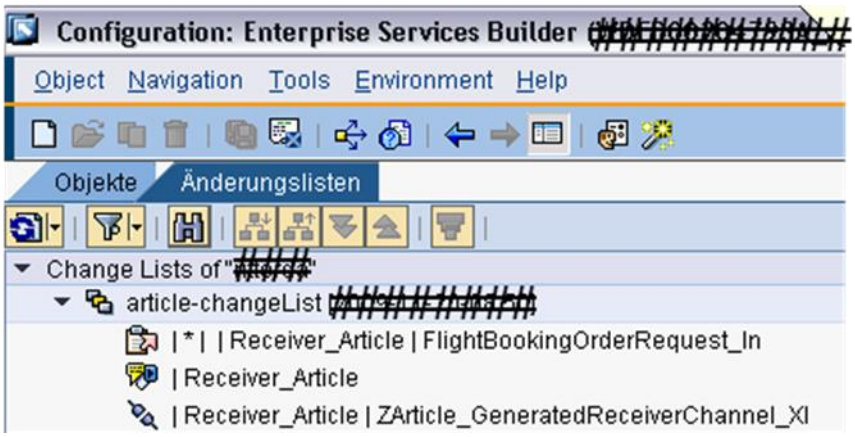

Figure 14: Integration Builder change list view showing the generated change list containing the business component, communication channel and receiver agreement

Why does the printout of the sample application not show any of the created components? That depends on the change list. If the change list has not been activated the `ReadContextCode.ACTIVE` setting prevents from retrieving the created but not yet activated change list objects. Solutions might be settings the context to `ReadContextCode.USER`, or programmatically activate the change list.

Appendix

Listing 1: DirectoryApiConfiguration

```
package com.sap.pi.directory.article;

/**
 * @author SAP AG
 *
 */

public class DirectoryApiConfiguration {
	private static String USERID = "youruser";
	private static String PASSWORD = "yourpassword";
	private static String SERVER_NAME = "yourservername";
	private static String SERVER_PORT = "yourserverport";

	/**
	 * @return
	 */
	public static String getUserid() {
		return USERID;
	}

	/**
	 * @return
	 */
	public static String getPassword() {
		return PASSWORD;
	}

	/**
	 * @return
	 */
	public static String getSERVER_NAME() {
		return SERVER_NAME;
	}

	/**
	 * @return
	 */
	public static String getSERVER_PORT() {
		return SERVER_PORT;
	}
}
```

Listing 2: DirectoryApiDAO

```
package com.sap.pi.directory.article;

import java.net.MalformedURLException;
```

```
import java.util.List;
import java.util.Map;

import javax.xml.ws.BindingProvider;

import com.sap.xi.basis.BusinessComponent;
import com.sap.xi.basis.BusinessComponentCreateChangeIn;
import com.sap.xi.basis.BusinessComponentIn;
import com.sap.xi.basis.BusinessComponentInService;
import com.sap.xi.basis.BusinessComponentQueryIn;
import com.sap.xi.basis.BusinessComponentQueryOut;
import com.sap.xi.basis.BusinessComponentReadIn;
import com.sap.xi.basis.BusinessComponentReadOut;
import com.sap.xi.basis.BusinessSystem;
import com.sap.xi.basis.BusinessSystemCreateChangeIn;
import com.sap.xi.basis.BusinessSystemIn;
import com.sap.xi.basis.BusinessSystemInService;
import com.sap.xi.basis.BusinessSystemQueryIn;
import com.sap.xi.basis.BusinessSystemQueryOut;
import com.sap.xi.basis.BusinessSystemReadIn;
import com.sap.xi.basis.ChangeListCreateOut;
import com.sap.xi.basis.ChangeListIn;
import com.sap.xi.basis.ChangeListInService;
import com.sap.xi.basis.CommunicationChannel;
import com.sap.xi.basis.CommunicationChannelCreateChangeIn;
import com.sap.xi.basis.CommunicationChannelDirection;
import com.sap.xi.basis.CommunicationChannelID;
import com.sap.xi.basis.CommunicationChannelIn;
import com.sap.xi.basis.CommunicationChannelInService;
import com.sap.xi.basis.CommunicationChannelQueryIn;
import com.sap.xi.basis.CommunicationChannelQueryOut;
import com.sap.xi.basis.CommunicationChannelReadIn;
import com.sap.xi.basis.CommunicationChannelReadOut;
import com.sap.xi.basis.CommunicationComponentID;
import com.sap.xi.basis.ConfigurationObjectModifyOut;
import com.sap.xi.basis.DesignObjectID;
import com.sap.xi.basis.GenericProperty;
import com.sap.xi.basis.LogMessageCollection;
import com.sap.xi.basis.MessageHeaderID;
import com.sap.xi.basis.ReadContextCode;
import com.sap.xi.basis.ReceiverAgreement;
import com.sap.xi.basis.ReceiverAgreementCreateChangeIn;
import com.sap.xi.basis.ReceiverAgreementIn;
import com.sap.xi.basis.ReceiverAgreementInService;
import com.sap.xi.basis.ReceiverAgreementQueryIn;
import com.sap.xi.basis.ReceiverAgreementQueryOut;
import com.sap.xi.basis.ReceiverAgreementReadIn;
import com.sap.xi.basis.ReceiverAgreementReadOut;
import com.sap.xi.basis.RestrictedBusinessComponent;
import com.sap.xi.basis.RestrictedBusinessSystem;
import com.sap.xi.basis.RestrictedChangeListID;
import com.sap.xi.basis.RestrictedCommunicationChannel;
import com.sap.xi.basis.RestrictedReceiverAgreement;
import com.sap.xi.basis.RestrictedSenderAgreement;
import com.sap.xi.basis.SenderAgreement;
import com.sap.xi.basis.SenderAgreementCreateChangeIn;
import com.sap.xi.basis.SenderAgreementIn;
import com.sap.xi.basis.SenderAgreementInService;
import com.sap.xi.basis.SenderAgreementQueryIn;
```

```
import com.sap.xi.basis.SenderAgreementQueryOut;
import com.sap.xi.basis.SenderAgreementReadIn;
import com.sap.xi.basis.SenderAgreementReadOut;
import com.sap.xi.basis.global.LONGDescription;

/**
 * @author SAP AG
 *
 */
public class DirectoryApiDAO {
      // BUSINESS SYSTEM

      /**
       * @return
       */
      public final List<CommunicationComponentID> queryBusinessSystemIn() {

            List<CommunicationComponentID> list = null;

            try {
                  // get web service
                  BusinessSystemInService service = new
BusinessSystemInService();

                  // get port
                  BusinessSystemIn systemComp =
service.getBusinessSystemInPort();

                  // Create a new instance of request object expected by
Communication
                  BusinessSystemQueryIn query = new BusinessSystemQueryIn();

                  // setup context and authorization
                  this.setupContext((javax.xml.ws.BindingProvider) systemComp,
                              DirectoryApiConfiguration.getUserid(),
                              DirectoryApiConfiguration.getPassword());

                  // action: use businessComp to query
                  BusinessSystemQueryOut result = systemComp.query(query);

                  LogMessageCollection logMsgs =
result.getLogMessageCollection();

                  int size = logMsgs.getLogMessageBusinessSystem().size();

                  for (int i = 0; i < size; i++) {

      System.out.println(logMsgs.getLogMessageBusinessSystem().get(i)
                                    .getLogMessageItem().getMessage());
                  }

                  list = result.getBusinessSystemID();

            } catch (MalformedURLException e) {
                  e.printStackTrace();
            }

            return list;
      }
```

```
	/**
	 * @param ids
	 * @return
	 */
	public final List<BusinessSystem> readBusinessSystemIn(
			List<CommunicationComponentID> ids) {

		List<BusinessSystem> businessSystem = null;

		try {
			// get web service
			BusinessSystemInService service = new
BusinessSystemInService();

			// get port
			BusinessSystemIn systemComp =
service.getBusinessSystemInPort();

			// Create a new instance of request object expected by
Communication
			BusinessSystemReadIn readIn = new BusinessSystemReadIn();
			readIn.setReadContext(ReadContextCode.ACTIVE);
			readIn.getBusinessSystemID().addAll(ids);

			// setup context and authorization
			this.setupContext((javax.xml.ws.BindingProvider) systemComp,
					DirectoryApiConfiguration.getUserid(),
					DirectoryApiConfiguration.getPassword());

			// action: use businessComp to query
			businessSystem = systemComp.read(readIn).getBusinessSystem();

		} catch (MalformedURLException e) {
			e.printStackTrace();
		}

		return businessSystem;
	}

	/**
	 * @param changeList
	 * @param masterLanguage
	 * @param id
	 */
	public final void createBusinessSystem(ChangeListCreateOut changeList,
			String masterLanguage, CommunicationComponentID id) {
		try {
			BusinessSystemInService service = new
BusinessSystemInService();
			BusinessSystemIn comp = service.getBusinessSystemInPort();

			// setup context and authorization
			this.setupContext((javax.xml.ws.BindingProvider) comp,
					DirectoryApiConfiguration.getUserid(),
					DirectoryApiConfiguration.getPassword());

			BusinessSystemCreateChangeIn create = new
BusinessSystemCreateChangeIn();
```

```
                create.setChangeListID(changeList.getChangeListID()
                            .getChangeListID());
                RestrictedBusinessSystem bs = new RestrictedBusinessSystem();
                bs.setMasterLanguage(masterLanguage);
                bs.setBusinessSystemID(id);
                create.getBusinessSystem().add(bs);

                ConfigurationObjectModifyOut out = comp.create(create);

                LogMessageCollection logMsgs = out.getLogMessageCollection();

                int size = logMsgs.getLogMessageBusinessSystem().size();

                for (int i = 0; i < size; i++) {

      System.out.println(logMsgs.getLogMessageBusinessSystem().get(i)
                                  .getLogMessageItem().getMessage());
                }

                if
(out.getLogMessageCollection().getLogMessageBusinessSystem()
                            .size() != 0) {
                      System.out.println(out.getLogMessageCollection()
                                  .getLogMessageBusinessSystem().get(0));
                }

          } catch (MalformedURLException e) {
                e.printStackTrace();
          }
      }

      // BUSINESS COMPONENT

      /**
       * @return
       */
      public final List<CommunicationComponentID> queryBusinessComponentIn() {
            List<CommunicationComponentID> list = null;
            try {
                // get service
                BusinessComponentInService service = new
BusinessComponentInService();

                // get port
                BusinessComponentIn businessComp = service
                            .getBusinessComponentInPort();

                // Create a new instance of request object expected by
Communication
                BusinessComponentQueryIn query = new
BusinessComponentQueryIn();

                // setup context and authorization
                this.setupContext((javax.xml.ws.BindingProvider) businessComp,
                            DirectoryApiConfiguration.getUserid(),
                            DirectoryApiConfiguration.getPassword());

                // action: use businessComp to query
                BusinessComponentQueryOut result = businessComp.query(query);
```

```
                LogMessageCollection logMsgs =
result.getLogMessageCollection();

                int size = logMsgs.getLogMessageBusinessComponent().size();

                for (int i = 0; i < size; i++) {

      System.out.println(logMsgs.getLogMessageBusinessComponent()
                                    .get(i).getLogMessageItem().getMessage());
                }

                list = result.getBusinessComponentID();

          } catch (MalformedURLException e) {
                e.printStackTrace();
          }

          return list;
      }

      /**
       * @param list
       * @return
       */
      public final List<BusinessComponent> readBusinessComponentIn(
                List<CommunicationComponentID> list) {
          List<BusinessComponent> businessComponents = null;

          try {
                // get service
                BusinessComponentInService service = new
BusinessComponentInService();

                // get port
                BusinessComponentIn businessComp = service
                            .getBusinessComponentInPort();

                // Create a new instance of request object expected by
Communication
                BusinessComponentReadIn read = new BusinessComponentReadIn();
                read.setReadContext(ReadContextCode.ACTIVE);
                read.getBusinessComponentID().addAll(list);

                // setup context and authorization
                this.setupContext((javax.xml.ws.BindingProvider) businessComp,
                            DirectoryApiConfiguration.getUserid(),
                            DirectoryApiConfiguration.getPassword());

                // action: use businessComp to query
                BusinessComponentReadOut result = businessComp.read(read);

                businessComponents = result.getBusinessComponent();

          } catch (MalformedURLException e) {
                e.printStackTrace();
          }

          return businessComponents;
```

```
	}

	/**
	 * @param changeList
	 * @param masterLanguage
	 * @param id
	 * @param inboundObjectId
	 * @param outboundObjectId
	 */
	public final void createBusinessComponent(ChangeListCreateOut changeList,
			String masterLanguage, CommunicationComponentID id,
			List<DesignObjectID> inboundObjectId,
			List<DesignObjectID> outboundObjectId) {
		try {
			BusinessComponentInService service = new
BusinessComponentInService();
			BusinessComponentIn comp =
service.getBusinessComponentInPort();

			// setup context and authorization
			this.setupContext((javax.xml.ws.BindingProvider) comp,
					DirectoryApiConfiguration.getUserid(),
					DirectoryApiConfiguration.getPassword());

			BusinessComponentCreateChangeIn create = new
BusinessComponentCreateChangeIn();
			create.setChangeListID(changeList.getChangeListID()
					.getChangeListID());
			RestrictedBusinessComponent bc = new
RestrictedBusinessComponent();
			bc.setMasterLanguage(masterLanguage);
			bc.setBusinessComponentID(id);
			bc.getInboundInterface().addAll(inboundObjectId);
			bc.getOutboundInterface().addAll(outboundObjectId);
			create.getBusinessComponent().add(bc);

			ConfigurationObjectModifyOut out = comp.create(create);

			LogMessageCollection logMsgs = out.getLogMessageCollection();
			int size = logMsgs.getLogMessageBusinessComponent().size();

			for (int i = 0; i < size; i++) {

	System.out.println(logMsgs.getLogMessageBusinessComponent()
					.get(i).getLogMessageItem().getMessage());
			}

		} catch (MalformedURLException e) {
			e.printStackTrace();
		}
	}

	// Sender AGREEMENT

	/**
	 * @return
	 */
	public final List<MessageHeaderID> querySenderAgreementIn() {
		List<MessageHeaderID> list = null;
```

```
            try {
                  // get service
                  SenderAgreementInService service = new
SenderAgreementInService();

                  // get port
                  SenderAgreementIn interfaceComp = service
                              .getSenderAgreementInPort();

                  // Create a new instance of request object expected by
Communication
                  SenderAgreementQueryIn query = new SenderAgreementQueryIn();

                  // setup context and authorization
                  this.setupContext((javax.xml.ws.BindingProvider)
interfaceComp,
                              DirectoryApiConfiguration.getUserid(),
                              DirectoryApiConfiguration.getPassword());

                  // action: use businessComp to query
                  SenderAgreementQueryOut result = interfaceComp.query(query);

                  LogMessageCollection logMsgs =
result.getLogMessageCollection();

                  int size = logMsgs.getLogMessageSenderAgreement().size();

                  for (int i = 0; i < size; i++) {

      System.out.println(logMsgs.getLogMessageSenderAgreement()
                                    .get(i).getLogMessageItem().getMessage());
                  }

                  list = result.getSenderAgreementID();

            } catch (MalformedURLException e) {
                  e.printStackTrace();
            }

            return list;
      }

      /**
       * @param list
       * @return
       */
      public final List<SenderAgreement> readSenderAgreementIn(
                  List<MessageHeaderID> list) {
            List<SenderAgreement> agrees = null;

            try {
                  // get service
                  SenderAgreementInService service = new
SenderAgreementInService();

                  // get port
                  SenderAgreementIn comp = service.getSenderAgreementInPort();
```

```
                // Create a new instance of request object expected by
Communication
                SenderAgreementReadIn read = new SenderAgreementReadIn();
                read.setReadContext(ReadContextCode.ACTIVE);
                read.getSenderAgreementID().addAll(list);

                // setup context and authorization
                this.setupContext((javax.xml.ws.BindingProvider) comp,
                            DirectoryApiConfiguration.getUserid(),
                            DirectoryApiConfiguration.getPassword());

                // action: use businessComp to query
                SenderAgreementReadOut result = comp.read(read);

                agrees = result.getSenderAgreement();

            } catch (MalformedURLException e) {
                e.printStackTrace();
            }

            return agrees;
      }

      /**
       * @param changeList
       * @param masterLanguage
       * @param header
       * @param channel
       * @param softwareComponentVersion
       */
      public final void createSenderAgreement(ChangeListCreateOut changeList,
                String masterLanguage, MessageHeaderID header,
                CommunicationChannelID channel, String
softwareComponentVersion) {
            try {
                SenderAgreementInService service = new
SenderAgreementInService();
                SenderAgreementIn comp = service.getSenderAgreementInPort();

                // setup context and authorization
                this.setupContext((javax.xml.ws.BindingProvider) comp,
                            DirectoryApiConfiguration.getUserid(),
                            DirectoryApiConfiguration.getPassword());

                SenderAgreementCreateChangeIn create = new
SenderAgreementCreateChangeIn();
                create.setChangeListID(changeList.getChangeListID()
                            .getChangeListID());

                RestrictedSenderAgreement senderAgreement = new
RestrictedSenderAgreement();

                senderAgreement.setMasterLanguage(masterLanguage);

                // sender agreement
                senderAgreement.setSenderAgreementID(header);

                // communication channel
                senderAgreement.setCommunicationChannel(channel);
```

```
                senderAgreement

      .setSoftwareComponentVersion(softwareComponentVersion);

                create.getSenderAgreement().add(senderAgreement);

                ConfigurationObjectModifyOut out = comp.create(create);

                LogMessageCollection logMsgs = out.getLogMessageCollection();

                int size = logMsgs.getLogMessageSenderAgreement().size();

                for (int i = 0; i < size; i++) {

      System.out.println(logMsgs.getLogMessageSenderAgreement()
                                    .get(i).getLogMessageItem().getMessage());
                }

            } catch (MalformedURLException e) {
                e.printStackTrace();
            }
      }

      // RECEIVER AGREEMENT

      /**
       * @return
       */
      public final List<MessageHeaderID> queryReceiverAgreementIn() {
            List<MessageHeaderID> list = null;

            try {
                // get service
                ReceiverAgreementInService service = new
ReceiverAgreementInService();

                // get port
                ReceiverAgreementIn interfaceComp = service
                            .getReceiverAgreementInPort();

                // Create a new instance of request object expected by
Communication
                ReceiverAgreementQueryIn query = new
ReceiverAgreementQueryIn();

                // setup context and authorization
                this.setupContext((javax.xml.ws.BindingProvider)
interfaceComp,
                            DirectoryApiConfiguration.getUserid(),
                            DirectoryApiConfiguration.getPassword());

                // action: use businessComp to query
                ReceiverAgreementQueryOut result = interfaceComp.query(query);

                LogMessageCollection logMsgs =
result.getLogMessageCollection();

                int size = logMsgs.getLogMessageReceiverAgreement().size();
```

```
                for (int i = 0; i < size; i++) {

      System.out.println(logMsgs.getLogMessageReceiverAgreement()
                                  .get(i).getLogMessageItem().getMessage());
                }

                list = result.getReceiverAgreementID();

          } catch (MalformedURLException e) {
                e.printStackTrace();
          }

          return list;
      }

      /**
       * @param list
       * @return
       */
      public final List<ReceiverAgreement> readReceiverAgreementIn(
                List<MessageHeaderID> list) {
          List<ReceiverAgreement> agrees = null;

          try {
                // get service
                ReceiverAgreementInService service = new
ReceiverAgreementInService();

                // get port
                ReceiverAgreementIn comp =
service.getReceiverAgreementInPort();

                // Create a new instance of request object expected by
Communication
                ReceiverAgreementReadIn read = new ReceiverAgreementReadIn();
                read.setReadContext(ReadContextCode.ACTIVE);
                read.getReceiverAgreementID().addAll(list);

                // setup context and authorization
                this.setupContext((javax.xml.ws.BindingProvider) comp,
                            DirectoryApiConfiguration.getUserid(),
                            DirectoryApiConfiguration.getPassword());

                // action: use businessComp to query
                ReceiverAgreementReadOut result = comp.read(read);

                agrees = result.getReceiverAgreement();

          } catch (MalformedURLException e) {
                e.printStackTrace();
          }

          return agrees;
      }

      /**
       * @param changeList
       * @param masterLanguage
       * @param header
```

```
	 * @param channelId
	 */
	public final void createReceiverAgreement(ChangeListCreateOut changeList,
			String masterLanguage, MessageHeaderID header,
			CommunicationChannelID channelId) {
		try {
			ReceiverAgreementInService service = new
ReceiverAgreementInService();
			ReceiverAgreementIn comp =
service.getReceiverAgreementInPort();

			// setup context and authorization
			this.setupContext((javax.xml.ws.BindingProvider) comp,
					DirectoryApiConfiguration.getUserid(),
					DirectoryApiConfiguration.getPassword());

			ReceiverAgreementCreateChangeIn create = new
ReceiverAgreementCreateChangeIn();
			create.setChangeListID(changeList.getChangeListID()
					.getChangeListID());

			RestrictedReceiverAgreement recAgreement = new
RestrictedReceiverAgreement();
			recAgreement.setMasterLanguage(masterLanguage);

			// receiver agreement id
			recAgreement.setReceiverAgreementID(header);

			// communication channel
			recAgreement.setCommunicationChannel(channelId);

			create.getReceiverAgreement().add(recAgreement);

			ConfigurationObjectModifyOut out = comp.create(create);

			LogMessageCollection logMsgs = out.getLogMessageCollection();

			int size = logMsgs.getLogMessageReceiverAgreement().size();

			for (int i = 0; i < size; i++) {

	System.out.println(logMsgs.getLogMessageReceiverAgreement()
					.get(i).getLogMessageItem().getMessage());
			}

		} catch (MalformedURLException e) {
			e.printStackTrace();
		}
	}

	// COMMUNICATION CHANNEL

	/**
	 * @return
	 */
	public final List<CommunicationChannelID> queryCommunicationChannelIn() {
		List<CommunicationChannelID> list = null;

		try {
```

```
                  // get service
                  CommunicationChannelInService service = new
CommunicationChannelInService();

                  // get port
                  CommunicationChannelIn interfaceComp = service
                              .getCommunicationChannelInPort();

                  // Create a new instance of request object expected by
Communication
                  CommunicationChannelQueryIn query = new
CommunicationChannelQueryIn();

                  // setup context and authorization
                  this.setupContext((javax.xml.ws.BindingProvider)
interfaceComp,
                              DirectoryApiConfiguration.getUserid(),
                              DirectoryApiConfiguration.getPassword());

                  // action: use businessComp to query
                  CommunicationChannelQueryOut result =
interfaceComp.query(query);

                  LogMessageCollection logMsgs =
result.getLogMessageCollection();

                  int size = logMsgs.getLogMessageCommunicationChannel().size();

                  for (int i = 0; i < size; i++) {

      System.out.println(logMsgs.getLogMessageCommunicationChannel()
                                    .get(i).getLogMessageItem().getMessage());
                  }

                  list = result.getCommunicationChannelID();

            } catch (MalformedURLException e) {
                  e.printStackTrace();
            }

            return list;
      }

      /**
       * @param list
       * @return
       */
      public final List<CommunicationChannel> readCommunicationChannelIn(
                  List<CommunicationChannelID> list) {

            List<CommunicationChannel> channel = null;

            try {
                  // get service
                  CommunicationChannelInService service = new
CommunicationChannelInService();

                  // get port
                  CommunicationChannelIn comp = service
```

```
                              .getCommunicationChannelInPort();

                  // Create a new instance of request object expected by
Communication
                  CommunicationChannelReadIn read = new
CommunicationChannelReadIn();
                  read.setReadContext(ReadContextCode.ACTIVE);
                  read.getCommunicationChannelID().addAll(list);

                  // setup context and authorization
                  this.setupContext((javax.xml.ws.BindingProvider) comp,
                              DirectoryApiConfiguration.getUserid(),
                              DirectoryApiConfiguration.getPassword());

                  // action: use businessComp to query
                  CommunicationChannelReadOut result = comp.read(read);

                  channel = result.getCommunicationChannel();

            } catch (MalformedURLException e) {
                  e.printStackTrace();
            }

            return channel;
      }

      public final String createCommunicationChannel(
                  ChangeListCreateOut changeList, String masterLanguage,
                  CommunicationChannelID channelID, DesignObjectID objectId,
                  CommunicationChannelDirection direction, String
transportProtocol,
                  String transportProtocolVersion, String messageProtocol,
                  String messageProtocolVersion, List<GenericProperty>
properties) {

            CommunicationChannelInService service;
            try {
                  service = new CommunicationChannelInService();
                  CommunicationChannelIn comp = service
                              .getCommunicationChannelInPort();

                  // setup context and authorization
                  this.setupContext((javax.xml.ws.BindingProvider) comp,
                              DirectoryApiConfiguration.getUserid(),
                              DirectoryApiConfiguration.getPassword());

                  CommunicationChannelCreateChangeIn create = new
CommunicationChannelCreateChangeIn();
                  create.setChangeListID(changeList.getChangeListID()
                              .getChangeListID());

                  RestrictedCommunicationChannel channel = new
RestrictedCommunicationChannel();

                  // master language
                  channel.setMasterLanguage(masterLanguage);

                  // communication channel
                  channel.setCommunicationChannelID(channelID);
```

```
            // adapter meta data
            channel.setAdapterMetadata(objectId);

            // channel data
            channel.setDirection(direction);

            channel.setTransportProtocol(transportProtocol);
            channel.setTransportProtocolVersion(transportProtocolVersion);
            channel.setMessageProtocol(messageProtocol);
            channel.setMessageProtocolVersion(messageProtocolVersion);

            // adapter specific attributes
            for (int i = 0; i < properties.size(); i++) {

channel.getAdapterSpecificAttribute().add(properties.get(i));
            }

            create.getCommunicationChannel().add(channel);

            ConfigurationObjectModifyOut out = comp.create(create);
            LogMessageCollection logMsgs = out.getLogMessageCollection();

            int size = logMsgs.getLogMessageCommunicationChannel().size();

            for (int i = 0; i < size; i++) {

System.out.println(logMsgs.getLogMessageCommunicationChannel()
                              .get(i).getLogMessageItem().getMessage());
            }

      } catch (MalformedURLException e) {
            e.printStackTrace();
      }

      return channelID.getChannelID();
}

/**
 * @param changeListName
 * @param changeListDescription
 * @param changeListDescriptionLanguageCode
 * @return
 */
public final ChangeListCreateOut createChangeList(String changeListName,
            String changeListDescription,
            String changeListDescriptionLanguageCode) {
      ChangeListInService service;
      try {
            service = new ChangeListInService();
            ChangeListIn comp = service.getChangeListInPort();

            // setup context and authorization
            this.setupContext((javax.xml.ws.BindingProvider) comp,
                        DirectoryApiConfiguration.getUserid(),
                        DirectoryApiConfiguration.getPassword());

            RestrictedChangeListID id = new RestrictedChangeListID();
            id.setName(changeListName);
```

```
            LONGDescription description = new LONGDescription();
            description.setValue(changeListDescription);

description.setLanguageCode(changeListDescriptionLanguageCode);
            id.setDescription(description);

            return comp.create(id);

      } catch (MalformedURLException e) {

            e.printStackTrace();
      }

      return null;
}

/**
 * @param changeList
 */
public final void activateChangeList(ChangeListCreateOut changeList) {
      try {
            ChangeListInService service = new ChangeListInService();

            ChangeListIn comp = service.getChangeListInPort();

            // setup context and authorization
            this.setupContext((javax.xml.ws.BindingProvider) comp,
                        DirectoryApiConfiguration.getUserid(),
                        DirectoryApiConfiguration.getPassword());

            LogMessageCollection logMsgs = comp.activate(changeList
                        .getChangeListID().getChangeListID());

            int size = logMsgs.getLogMessageChangeList().size();

            for (int i = 0; i < size; i++) {

System.out.println(logMsgs.getLogMessageChangeList().get(i)
                              .getLogMessageItem().getMessage());
            }
      } catch (MalformedURLException e) {
            e.printStackTrace();
      }
}

/**
 * @param changeList
 */
public final void rejectChangeList(ChangeListCreateOut changeList) {
      ChangeListInService service;
      try {
            service = new ChangeListInService();

            ChangeListIn comp = service.getChangeListInPort();

            // setup context and authorization
            this.setupContext((javax.xml.ws.BindingProvider) comp,
                        DirectoryApiConfiguration.getUserid(),
```

```
                              DirectoryApiConfiguration.getPassword());

                  comp.revert(changeList.getChangeListID().getChangeListID());
            } catch (MalformedURLException e) {
                  e.printStackTrace();
            }
      }

      /**
       * @param bindingProvider
       * @param user
       * @param password
       */
      private final void setupContext(BindingProvider bindingProvider,
                  String user, String password) {

            javax.xml.ws.BindingProvider bp = bindingProvider;
            Map<String, Object> context = bp.getRequestContext();
            StringBuffer sb = new StringBuffer(
                        "Request context of service endpoint");
            if (context == null) {
                  System.out.println(sb.append(" is null").toString());
            } else {
                  String endpointAddress = (String) context
                              .get(BindingProvider.ENDPOINT_ADDRESS_PROPERTY);
                  String endPointPrefix = "http://";
                  String addressTailWithHostAndPort = endpointAddress
                              .substring(endPointPrefix.length());
                  endpointAddress = new StringBuffer(endPointPrefix).append(

      DirectoryApiConfiguration.getSERVER_NAME()).append(
                              ":" + DirectoryApiConfiguration.getSERVER_PORT())
                              .append(
                                          addressTailWithHostAndPort

      .substring(addressTailWithHostAndPort

      .indexOf("/"))).toString();

                  context.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
                              endpointAddress);
                  context.put(BindingProvider.USERNAME_PROPERTY, user);
                  context.put(BindingProvider.PASSWORD_PROPERTY, password);
            }
      }
}
```

Listing 3: DirectoryAPIMain

```
package com.sap.pi.directory.article;

import java.util.ArrayList;
import java.util.List;
```

```
import com.sap.xi.basis.BusinessComponent;
import com.sap.xi.basis.BusinessSystem;
import com.sap.xi.basis.ChangeListCreateOut;
import com.sap.xi.basis.CommunicationChannel;
import com.sap.xi.basis.CommunicationChannelDirection;
import com.sap.xi.basis.CommunicationChannelID;
import com.sap.xi.basis.CommunicationComponentID;
import com.sap.xi.basis.DesignObjectID;
import com.sap.xi.basis.GenericProperty;
import com.sap.xi.basis.MessageHeaderID;
import com.sap.xi.basis.ReceiverAgreement;
import com.sap.xi.basis.SenderAgreement;

public class DirectoryAPIMain {

	/**
	 * @param args
	 */
	public static void main(String[] args) {

		// data access object
		DirectoryApiDAO directory = new DirectoryApiDAO();
		String masterLanguage = "EN";

		// --- W R I T E --- //

		// create variables
		String businessComponent = "Receiver_Article";
		String receiverChannelIdString =
"ZArticle_GeneratedReceiverChannel_XI";

		// create change list
		ChangeListCreateOut changeList = directory.createChangeList(
				"article-changeList",
				"Change list for article on Integration Directory API",
masterLanguage);

		// create new receiver business component - if it does not exist
		CommunicationComponentID id = new CommunicationComponentID();
		id.setComponentID(businessComponent); // does not need to exist in
SLD

		List<DesignObjectID> inboundObjectId = new
ArrayList<DesignObjectID>();
		List<DesignObjectID> outboundObjectId = new
ArrayList<DesignObjectID>();

		directory.createBusinessComponent(changeList, masterLanguage, id,
				inboundObjectId, outboundObjectId);

		// create RECEIVER communication channel for business system
		CommunicationChannelID receiverChannelID = new
CommunicationChannelID();
		//receiverChannelID.setPartyID("");
		receiverChannelID.setComponentID(businessComponent); //
prerequisite: is created
		receiverChannelID.setChannelID(receiverChannelIdString);
```

```
            // adapter meta data
            DesignObjectID objectId = new DesignObjectID();
            objectId.setName("XI");
            objectId.setNamespace("http://sap.com/xi/XI/System");
            objectId.setSoftwareComponentVersionID("e5a519a0-7661-11dc-a2e8-
e95d0a115642");

            // channel data
            CommunicationChannelDirection direction =
CommunicationChannelDirection.RECEIVER;

            String transportProtocol = "HTTP";
            String transportProtocolVersion = "1.0";
            String messageProtocol = "XI";
            String messageProtocolVersion = "3.0";

            // adapter specific attributes
            List<GenericProperty> properties = new ArrayList<GenericProperty>();

            GenericProperty property = new GenericProperty();
            property.setName("host");
            property.setValue("TODO: set your host name");
            properties.add(property);

            property = new GenericProperty();
            property.setName("addressingMode");
            property.setValue("url");
            properties.add(property);

            property = new GenericProperty();
            property.setName("port");
            property.setValue("TODO: set your port");
            properties.add(property);

            property = new GenericProperty();
            property.setName("path");
            property.setValue("/sap/xi/engine?type=entry");
            properties.add(property);

            String receiverChannel = directory.createCommunicationChannel(
                        changeList, masterLanguage, receiverChannelID, objectId,
                        direction, transportProtocol, transportProtocolVersion,
                        messageProtocol, messageProtocolVersion, properties);

            // create receiver agreement
            // receiver agreement id
            MessageHeaderID header = new MessageHeaderID();
            header.setSenderComponentID("*");
            header.setInterfaceName("FlightBookingOrderRequest_In");
            header.setInterfaceNamespace("http://sap.com/xi/XI/Demo/Airline");
            header.setReceiverComponentID(businessComponent);

            // communication channel
            CommunicationChannelID channelId = new CommunicationChannelID();
            //channelId.setPartyID("");
            channelId.setChannelID(receiverChannel);
            channelId.setComponentID(businessComponent);
```

```
            directory.createReceiverAgreement(changeList, masterLanguage,
header,
                        channelId);

            // activate changes if required
            //directory.activateChangeList(changeList);

            // --- R E A D --- //

            // get communictation components
            List<BusinessSystem> bs = directory.readBusinessSystemIn(directory
                        .queryBusinessSystemIn());

            System.out.println("Business Systems:");
            for (int i = 0; i < bs.size(); i++) {
                  // display party and id

      System.out.println(bs.get(i).getBusinessSystemID().getPartyID()
                              + "#" +
bs.get(i).getBusinessSystemID().getComponentID());
            }

            System.out.println("#------------------#");

            List<BusinessComponent> bc = directory

      .readBusinessComponentIn(directory.queryBusinessComponentIn());

            System.out.println("Business Components:");
            for (int i = 0; i < bc.size(); i++) {
                  // display party and id
                  System.out

      .println(bc.get(i).getBusinessComponentID().getPartyID()
                                          + "#"
                                          + bc.get(i).getBusinessComponentID()
                                                      .getComponentID());
            }

            // get sender agreements
            System.out.println("#------------------#");

            List<SenderAgreement> sa = directory.readSenderAgreementIn(directory
                        .querySenderAgreementIn());

            System.out.println("Sender Agreements:");
            for (int i = 0; i < sa.size(); i++) {
                  // display id
                  System.out.println(sa.get(i).getSenderAgreementID()
                              .getSenderComponentID());
            }

            // get receiver agreements
            System.out.println("#------------------#");

            List<ReceiverAgreement> ra = directory

      .readReceiverAgreementIn(directory.queryReceiverAgreementIn());
```

```
            System.out.println("Receiver Agreements:");
            for (int i = 0; i < ra.size(); i++) {
                  // display id
                  System.out.println(ra.get(i).getReceiverAgreementID()
                              .getReceiverComponentID());
            }

            // get communication channels
            System.out.println("#------------------#");

            List<CommunicationChannel> cc = directory
                        .readCommunicationChannelIn(directory
                                    .queryCommunicationChannelIn());

            System.out.println("Communication Channels:");
            for (int i = 0; i < cc.size(); i++) {
                  // display id
                  System.out.println(cc.get(i).getCommunicationChannelID()
                              .getChannelID());
            }
      }
}
```

Related Content

[1] Integration Directory API.
http://help.sap.com/saphelp_nwpi71/helpdata/en/46/6dca42e5c269dfe10000000a11466f/frameset.htm

[2] SAP NetWeaver Process Integration 7.1. http://www.sdn.sap.com/irj/sdn/nw-pi71;jsessionid=(J2EE3414700)ID0801759550DB009548406612283327 15End.

[3] Li, William: Directory API Development.
http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/11655%3Fpage%3Dlast%26x-showcontent%3Doff%26x-maxdepth%3D0. 20.10.2008.

Copyright