SeeBeyond ICAN Suite

SWIFT Alliance Gateway eWay Intelligent Adapter User's Guide

Release 5.0

Version 20041203124411.

Contents

Chapter 5

Chapter 6

Chapter 1

Introduction

This guide explains how to install, use, and operate the SeeBeyond® Integrated Composite Application Network Suite™ (ICAN) SWIFT Alliance Gateway eWay Intelligent Adapter, referred to as the SWIFT AG eWay throughout this guide.

This chapter provides a brief overview of operations, components, and general features of SWIFT Alliance Gateway and the eWay, as well as an introduction to the document.

What's in This Chapter

1.1 About SWIFT Alliance Gateway

This section provides an overview of the Society for Worldwide Interbank Financial Telecommunication (SWIFT) and how SWIFT Alliance Gateway operates.

1.1.1 Introduction to SWIFT

SWIFT is a bank-owned cooperative that supplies secure payment event transfer, matching, and other services to owner/member banks and other financial organizations (including brokers, securities deposit and clearing organizations, and stock exchanges) using its SWIFT Transport Network (STN).

The types of messages processed by SWIFT include:

- **Payments**: Clearing and settlements between member banks.
- **Securities**: Clearing and settlements and cross border electronic trade confirmations.
- **Forex, Money Markets and Derivatives**: Confirmation of trades, marketing and reporting facilities.
- **Trade Finance**: Documenting credits and collections.

1.1.2 Introduction to SWIFT Alliance Gateway

SWIFT Alliance Gateway provides application-to-application communication over a secure IP network (SIPN). The gateway is a single point of entry into the SIPN. It is used to exchange messages between applications. Most message exchange is accomplished using the SWIFT InterAct Service or the SWIFT FileAct Service.

Note: *See the **SWIFT Alliance Gateway Operations Guide** for complete details on the operation of SWIFT Alliance Gateway.*

Application Connectivity

Applications can connect to SWIFT Alliance Gateway in a variety of ways. These connections are called interfaces into SWIFT Alliance Gateway. SWIFT provides its own remote application programming interface (API) that allows customer applications to communicate over the SIPN.

Legacy applications have another option, the MQ-Series API, that is used to exchange messages. Finally, SWIFT provides the SWIFT Alliance WebStation as a stand-alone program for configuring SWIFT Alliance Gateway. However, while there are many ways to communicate using SWIFT Alliance Gateway, the operations employed in the communication process remain the same.

SWIFT Messaging Operation

SWIFT Alliance Gateway allows you to configure how messages are routed based on:

- Correspondents
- Message Partners
- Endpoints

Correspondents

The primary concept within SWIFT Alliance Gateway is the Correspondent. A Correspondent is a representation of the entity sending the message. A Correspondent can be an institution, a department, a person, an application, or a piece of software. The Correspondent can be internal or external to your particular institution.

An example of an external Correspondent could be a bank with which your institution exchanges data. An internal Correspondent could be a logging operation sending administrative messages within one of your own systems.

Message Partner

Each Correspondent has an associated Message Partner. A Message Partner is an application that sends and receives messages on behalf of the Correspondent. The Message Partner is used internally by SWIFT Alliance Gateway.

The Message Partner is configured with information used for messaging exchanges. This information can be, for example, the exact application a correspondent wants to communicate with, the format of the messages exchanged, or the employed interface (such as, Remote API, MQ-Series API, or another API) used to complete the exchange.

Endpoint

An Endpoint maps the interfaces of one application to another. For example, an Endpoint can be set up between Message Partner 1, which communicates using the Remote API, and Message Partner 2, which communicates using the MQ-Series API.

Messages that enter SWIFT Alliance Gateway can be routed based on the following criteria:

- Correspondents that sent the message
- Predefined Endpoints
- Format of the message

SWIFT Messaging Services

The SWIFT AG eWay provides secure messaging services (both receiving and transmitting) between SWIFT financial institutions. The SWIFT AG eWay is designed specifically to interface with SWIFT Alliance Gateway and enables the SeeBeyond eGate Integrator (or eInsight ESB) system to exchange data with SWIFT Alliance by providing:

- **Automated Integration** of securities messages in the new securities standards, which are based on the ISO15022 Data Dictionary.
- **Translation** of incoming messages received from SWIFT into the format required by existing applications.
- **Security**, by being subject to the same authentication features as other SWIFT Alliance components.

1.2 About the SWIFT AG eWay

The eWay provides connectivity to SWIFT Alliance Gateway using the SWIFT Remote API and allows the synchronous and asynchronous sending of messages through this interface.

For its general operation, the eWay uses the following Object Type Definitions (OTDs):

- **SAGApplication**
- **SAGProcessControlApplication**

See **Chapter 4** for specific information on the eWay's OTDs. See the *eGate Integrator User's Guide* for more general information on eGate OTDs.

You can set up the SWIFT AG eWay using the eGate Java-based Collaboration Editor or the Business Process Execution Language (BPEL) interface, in eGate with eInsight Business Process Manager or eInsight ESB.

See the *eInsight Business Process Manager User's Guide* for more information on the BPEL interface.

1.2.1 SAGApplication OTD

The natural point of connection for the eWay is the **SAGApplication** OTD. Using either the Java-based Collaboration Editor or the BPEL Business Processes, this OTD defines a generic method for connecting to SWIFT Alliance Gateway.

Java-based Methods

In the Java-based Collaborations, the **SAGApplication** OTD defines generic request-and-response messages. The request message defines a set of attributes that must be filled in. Most of these attributes are used in routing and security. The payload of the request message is the actual data the receiving application uses.

A server mode also exists for the Java-based Collaboration, allowing request messages to be received and response messages to be returned.

BPEL Business Processes

In BPEL, the same functionality exists as in the Java-based Collaboration setup. However, the eWay provides the method capabilities as Business Process operations, for example:

- **send**
- **sendRequest**
- **retrieveResponse**

These operations power the eWay's eInsight Business Processes to perform the same basic operations as those available in the Java-based Collaborations.

1.2.2 SAGProcessControlApplication OTD

The SWIFT AG eWay also provides the **SAGProcessControlApplication** OTD. This OTD is designed to provide administrative control over SWIFT Alliance Gateway. Communication with the SWIFT Alliance Gateway happens basically as set of commands exposed as Java-based methods on the OTD.

See the **Javadoc** for a more detailed description of the methods used in this OTD. **Chapter 6** provides an explanation of the eWay's **Javadoc** and how to use it.

Java-based Methods

In Java-based Collaborations, the **SAGProcessControlApplication** OTD defines a set of commands you can invoke to control the SWIFT Alliance Gateway. This OTD provides the following key attributes that are generic across all commands:

- Operator name
- Operator password

The possible commands include **start**, **stop**, **validate**, **retrieveStatus**, **refresh**, **traceSet**, **traceReset**, **checkIntegrity**, **backup**, **readLog**, and **archive**.

The **SAGProcessControlApplication** OTD only exists in the client mode.

BPEL Business Processes

In BPEL, the same commands represented as methods in the Java-based Collaborations are represented as operations, that is, eInsight Business Processes. All of the Business Process operations are stateless.

Each input message contains all the required information to correctly perform the command required by the message's associated Business Process.

1.2.3 eWay General Operation

You can implement and operate the SWIFT AG eWay with eGate Integrator (with or without eInsight) or eInsight ESB. Either implementation lets the eWay operate within the functionality of the ICAN Suite. eInsight ESB only allows for BPEL implementation of the eWay.

1.2.4 Components of the eWay

The SWIFT AG eWay includes the following components:

- The **SwiftAGeWay.sar** file, which when installed, contains the SWIFT AG eWay.

  See **Chapter 2** for details on installation.

- A default configuration template

  See **Chapter 3** for details.

- The following OTDs to implement the functionality of the eWay:
  - **SAGApplication**
  - **SAGProcessControlApplication**

  See **Chapter 4** for more information.

- A **sagjni.jar** file, which must be added to the Integration Server classpath, and one of three shared libraries, **stcsagjni.dll** (for Windows), **libsagjni.so** (for Solaris), and **libsagjni.a** (for AIX), which must be added to the Integration Server library path.

See Chapter 2 for details.

- Two sample eWay implementation Project files (Java-based and BPEL), a **Javadoc** file, and a **Readme.txt** file for the eWay. Input data is included with the samples.

  See Chapter 5 for details on the sample Project files. See Chapter 6 for an introduction to the **Javadoc**. Chapter 2 explains how to download the **Readme.txt** file

1.3 About This Document

This section explains information about this eWay user's guide.

1.3.1 What's in This Document

This document provides information about installing, configuring, and using the SWIFT AG eWay and includes the following chapters:

- Chapter 1 **"Introduction"** provides an overview of SWIFT messaging and SWIFT Alliance Gateway, as well as the SWIFT AG eWay and the guide.
- Chapter 2 **"Installing the eWay"** provides the supported operating systems and system requirements for the SWIFT AG eWay. It also includes directions for installing the SWIFT AG eWay and additional files, and accessing the accompanying documentation and sample Projects.
- Chapter 3 **"Configuring the eWay"** explains the process of configuring the SWIFT AG eWay properties, allowing the eWay to run in the current environment.
- Chapter 4 **"Using the eWay OTDs"** generally describes and explains the OTDs used with the eWay, including their Java classes; it also summarizes their BPEL functionality.
- Chapter 5 **"Reviewing the Sample Projects"** describes the implementation and functionality of the SWIFT AG eWay using the eGate Integrator, as well as eInsight (or eInsight ESB only for BPEL) with the eWay's sample Projects.
- Chapter 6 **"Using eWay Java Methods"** describes the SWIFT AG eWay Java classes and provides directions for accessing the SWIFT AG eWay **Javadoc**.

1.3.2 Scope

This guide describes and explains how to install and use the SWIFT AG eWay with eGate, to function within the ICAN Suite of products. Additional detailed information, such as detailed steps required to create sample integration Projects are not included in this guide. However, sample Projects are described and reviewed, to demonstrate how to implement the SWIFT AG eWay in typical environments.

1.3.3 Intended Audience

This guide is intended for computer users who have the ability and responsibility of setting up and maintaining a fully functioning ICAN Suite system. These persons must also understand any operating systems on which the current ICAN Suite is installed, for example Windows or Solaris UNIX, and must be thoroughly familiar with Windows-style user interface operations.

1.3.4 Document Conventions

The following conventions are observed throughout this document.

Table 1 Document Conventions

Text	Convention	Example
Names of buttons, files, icons, parameters, variables, methods, menus, and objects	**Bold** text	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**. ▪ Select the **logicalhost.exe** file. ▪ Enter the **timeout** value. ▪ Use the **getClassName()** method. ▪ Configure the **Inbound** File eWay.
Command line arguments, code samples	`Fixed` font. Variables are shown in ***`bold italic`***.	`bootstrap -p` ***`password`***
Hypertext links	**Blue** text	See **"Document Conventions" on page 12**
Hypertext links for Web addresses (URLs) or email addresses	**Blue underlined** text	**http://www.seebeyond.com** **docfeedback@seebeyond.com**

1.3.5 Screenshots

Depending on what products you have installed, and how they are configured, the screenshots in this document may differ from what you see on your system.

1.4 Related Documents

Use the following related SeeBeyond guides as a reference for additional information in using the SWIFT AG eWay:

- *ICAN Suite Installation Guide*
- *eGate Integrator User's Guide*
- *eGate Integrator System Administration Guide*
- *eInsight Business Process Manager User's Guide*
- *eInsight ESB User's Guide*

1.5 References

Use the following related third-party guides as a reference for additional information on SWIFT Alliance Gateway:

- SWIFT Web site
- *SWIFT Alliance Gateway Getting Started*
- *SWIFT Alliance Gateway Installation Guide*
- *SWIFT Alliance Gateway Operations Guide*
- *SWIFT Remote APIs Installation Guide*

1.6 SeeBeyond Web Site

The SeeBeyond Web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.seebeyond.com

1.7 SeeBeyond Documentation Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

docfeedback@seebeyond.com

Chapter 2

Installing the eWay

This chapter explains how to install the SWIFT AG eWay, as well as supported operating systems and system requirements. The chapter also includes necessary post-installation procedures.

What's in This Chapter

2.1 Supported Operating Systems

The SWIFT AG eWay is available for the following operating systems:

- Windows 2000
- IBM AIX 5.1L
- Sun Solaris 8

2.2 System Requirements

To use the SWIFT AG eWay, you need:

- eGate Logical Host
- TCP/IP network connection

Logical Host requirements

The eWay must have its configuration properties set and be administered using the Enterprise Designer. For complete information on the Enterprise Designer system requirements, see the *ICAN Suite Installation Guide*.

2.3 Supported External Applications

The SWIFT AG eWay supports SWIFT Alliance Gateway version 4.0.20.

To operate the eWay, you must use SWIFT Alliance Gateway, version 4.0.20, and SWIFT Remote APIs, version 4.0.20.

Optionally you may also use SWIFT Net Link, version 4.4.20.

See the *SWIFT Alliance Gateway Installation Guide* and the *SWIFT Remote APIs Installation Guide* on the SWIFT Alliance Gateway installation CD-ROM for complete information on how to install and configure this software.

2.4 Before You Install

Open and review the **Readme.txt** file for the ICAN Suite (located in the root directory of the ICAN installation's Repository CD-ROM) for any current information you may need, for example for eGate or eInsight, before installing the eWay.

Note: *See the* ***SeeBeyond ICAN Suite Installation Guide*** *for details.*

Also, the SWIFT AG eWay has its own **Readme.txt** file with additional information specific to the eWay. Later sections in this chapter explain how to obtain this file.

2.5 Installing the eWay Product Files

During the ICAN Suite installation operation, the Enterprise Manager, a Web-based application, is used to select and upload eWay and add-on files (.**sar** files) from the ICAN installation CD-ROM to the Repository.

When the Repository is running on a UNIX operating system, eWays are loaded using the Enterprise Manager on a Windows computer connected to the Repository server, using Internet Explorer.

Installing the SWIFT AG eWay on an eGate-supported System

The SWIFT AG eWay can be installed during the installation of eGate. The eGate installation process includes the following operations:

- Installing the eGate Repository
- Uploading products to the Repository
- Downloading the components (including the eGate Enterprise Designer and the Logical Host)
- Viewing the product information home pages

Note: *You can install the SWIFT AG eWay solely with eGate, with eGate and eInsight, or solely with eInsight ESB.*

Follow the instructions for installing the ICAN Suite found in the *SeeBeyond ICAN Suite Installation Guide*, and include the following steps:

1. After the eGate or eInsight core products are uploaded to the Repository using the Enterprise Manager, select and upload the **FileeWay.sar**. The File eWay is used by the eWay's Project sample. You must upload the File eWay (**FileeWay.sar**.) before uploading the SWIFT AG eWay (**SwiftAGeWay.sar**).
2. After the File eWay is uploaded, upload **SwiftAGeWay.sar** to install the SWIFT AG eWay.
3. Next, upload the **SWIFTAGeWayDocs.sar**. This file contains:
   - The eWay user's guide as a .**pdf** file
   - The **Javadoc** (.**zip**) file
   - A **.zip** file containing the sample Project files (see **Chapter 5** for details on these files)
   - A **Readme.txt** file.

   To obtain these files, follow the instructions provided by the Enterprise Manager.
4. If needed, continue installing eGate and/or additional ICAN Suite products as instructed in the *SeeBeyond ICAN Suite Installation Guide.*

2.6 After You Install

Once the SWIFT AG eWay is installed and configured, it must then be incorporated into a Project before it can perform its intended functions. See the *eGate Integrator User's Guide* for more information on incorporating the eWay into an eGate Project.

The rest of this section provides important information on steps you must take to ensure that the eWay operates correctly with eGate, as well as eInsight (or eInsight ESB).

2.6.1 Configuring the JNI Portion of the SWIFT AG eWay

After uploading the **SwiftAGeWay.sar** file, click the **Downloads** tab of the Enterprise Manager. A new component, **SWIFT Alliance Gateway (SAG) Component**, is now available.

This component provides OS-specific code and JNI-wrapper code, which allow the SWIFT AG eWay to connect properly with SWIFT Alliance Gateway.

To download the SWIFT Alliance Gateway (SAG) Component

1. Click the component's name in the same way as you did other eGate components.

   The .**zip** file window opens displaying the component's contents.

2 Unzip the resulting .**zip** file to a temporary folder under your ICAN installation folder.

Adding the .jar file to the Integration Server Classpath

The eWay's Java JNI file (**sagjni.jar**) can be added to the Integration Server classpath after you have created an eGate Environment and added a Logical Host for your Project.

To add the .jar file to the Integration Server classpath

1 From the **Environment Explorer** tree, right-click the **Logical Host** and select **Upload File** from the shortcut menu.

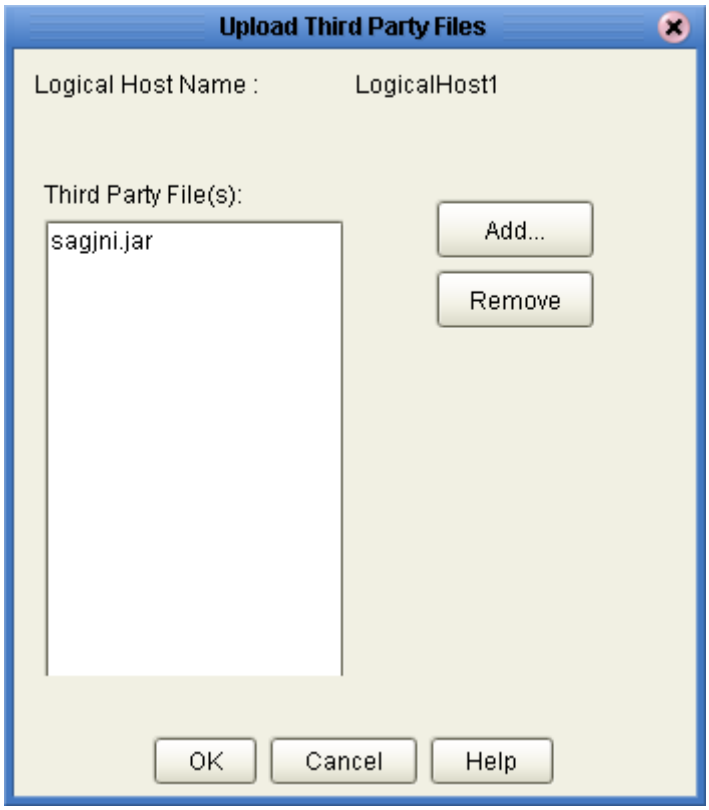
The **Upload Third Party Files** dialog box appears.

2 Click **Add**.

3 Browse to the temporary directory to which you downloaded the **.jar** file.

4 Add **sagjni.jar** to the **Third Party Files** field (see Figure 1).

Figure 1 Upload Third Party Files Dialog Box



5 When you are finished, click **OK**.

Adding Shared Library Files to Integration Server Library Paths

You must copy the appropriate shared library file from your temporary folder where you downloaded each **SWIFT Alliance Gateway (SAG) Component** to the appropriate Integration Server shared library path.

The exact location where you paste the file depends on the operating system you are using. Also, you must update the appropriate environment variable to reflect the file's location.

The following procedures provide the file names, the correct path locations, and the methods for updating the environment variables:

To add the shared library .dll (stcsagjni.dll) file to the path for Windows

1. After you first run the Logical Host, copy the **stcsagjni.dll** file from the temporary directory to which you downloaded the file.
2. Paste the file to the following location:

   ```
   <ican>\logicalhost\stcis\lib
   ```

   Where `<ican>` is your installed ICAN Suite.
3. Update the Path Environment variable to include the `<ican>\logicalhost\stcis\lib` directory as follows:
   A. Right-click **My Computer** on your Desktop.
   B. Select **Properties**.

      The **System Properties** dialog box appears.
   C. Click the **Advanced** tab.
   D. Click **Environment Variables**.
   E. Under **System variables**, select the **Path** variable and click **Edit**.

      The **Edit System Variable** dialog box appears, including a text box, allowing you to edit the path variables.
   F. Enter the `<ican>\logicalhost\stcis\lib` directory as the last entry in the **Variable value** text box. Make sure to enter a semicolon first, to separate your entry from the previous entry in the text box.
   G. When you are finished, click **OK** to close the dialog box.
   H. Click **OK** to close each properties dialog box.

To add the shared library . (stcsagjni.so) file to the path for Solaris

1. After you first run the Logical Host, copy the **stcsagjni.so** file from the temporary directory to which you downloaded the file.
2. Paste the file to the following location:

   ```
   <ican>\logicalhost\stcis\lib
   ```

   Where `<ican>` is your installed ICAN Suite.

3 Update the global LD_LIBRARY_PATH environment variable to include the `<ican>/logicalhost/stcis/lib` directory as follows:

- For SH-based shells, including SH, BASH, and KSH, you must update the LD_LIBRARY_PATH variable in the **.profile** file to include the directory path, for example:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<ican>/
logicalhost/stcis/lib
```

- For CSH-based shells, including CSH and TCSH, you must update the LD_LIBRARY_PATH variable in the **.cshrc** file to include the directory path, for example:

```
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:<ican>/
logicalhost/stcis/lib
```

Note: *How this environment variable is set can vary depending on the specific shell you are using.*

To add the shared library . (stcsagjni.so) file to the path for AIX

1 After you first run the Logical Host, copy the **stcsagjni.so** file from the temporary directory to which you downloaded the file.

2 Paste the file to the following location:

```
<ican>\logicalhost\stcis\lib
```

Where `<ican>` is your installed ICAN Suite.

3 Update the global LIBPATH (library path) environment variable to include the `<ican>/logicalhost/stcis/lib` directory as follows:

- For SH-based shells, including SH, BASH, and KSH, you must update the LIBPATH variable in the .**profile** file to include the directory path, for example:

```
export LIBPATH=$LIBPATH:<ican>/logicalhost/stcis/lib
```

- For CSH-based shells, including CSH and TCSH, you must update the LIBPATH variable in the **.cshrc** file to include the directory path, for example:

```
setenv LIBPATH ${LIBPATH}:<ican>/logicalhost/stcis/lib
```

Note: *How this environment variable is set can vary depending on the specific shell you are using.*

2.6.2 Configuring Specific Parameters

For the SWIFT AG eWay to properly connect to SWIFT Alliance Gateway, certain specific parameters for SWIFT Alliance Gateway must be set in the environment.

Configuring SWNET_CFG_PATH

One such environment variable is the SWNET_CFG_PATH. This variable tells SWIFT Alliance Gateway where to look for configuration files. These files are generated and placed in this directory after the deployment of a Project that uses the SWIFT AG eWay.

To make the SWIFT Alliance Gateway configuration files available

1. Create a directory to hold the configuration files. A directory such as `<ican installation>/swnet_cfg_path` is recommended.
2. You must set the SWNET_CFG_PATH environment variable to point to this directory.

How the environment variable is set up depends on the operating system. The following procedures explain these setup operations:

To set up the environment variable in Windows

1. Right-click **My Computer** on your Desktop.
2. Select **Properties**.

   The **System Properties** dialog box appears.
3. Click the **Advanced** tab.
4. Click **Environment Variables**.
5. Under **User variables**, click **New**.

   The **New User Variables** dialog box appears, including a text box, allowing you to enter new user variables
6. Enter SWNET_CFG_PATH for the **Variable name** and `<ican installation>/swnet_cfg_path` for the **Variable value**.
7. When you are finished, click **OK** to close the dialog box.
8. Click **OK** to close each properties dialog box.

In Solaris and AIX, you must update the global SWNET_CFG_PATH environment variable to include the `<ican installation>/swnet_cfg_path` directory.

To set up the environment variable in Solaris and AIX

- For SH-based shells, including SH, BASH, and KSH, you must update the SWNET_CFG_PATH variable in the .**profile** file to include the directory path, for example:

```
export SWNET_CFG_PATH=<ican installation>/swnet_cfg_path
```

- For CSH-based shells, including CSH and TCSH, you must update the SWNET_CFG_PATH variable in the .**cshrc** file to include the directory path, for example:

```
setenv SWNET_CFG_PATH ${SWNET_CFG_PATH}:<ican
installation>/swnet_cfg_path
```

Note: *How this environment variable is set can vary depending on the specific shell you are using.*

Updating Environment Variables for Remote API Shared Libraries

Another environment variable that must be updated is the library path to the actual SWIFT Alliance Gateway Remote API shared libraries. The actual environment variable that must be updated depends on the operating system.

You normally update environment variables in eGate using the **Environment Variables** setting in the **Integration Server** properties dialog box. However, this procedure is *not* recommended for this variable. The **Environment Variables** property entered in this dialog box overwrites any existing settings for that variable.

The recommended approach in this case is to update the variable using the command line, before running the Logical Host. Depending on the operating system you are running on, use the following procedures:

To update the Remote API Shared Libraries environment variable for Windows

1. Identify the installation location of your SWIFT Remote API libraries. It is usually located at `<SWIFT Install directory>/RA/lib`.
2. Update the Path Environment variable to include the `<SWIFT Install directory>/RA/lib` directory as follows:
   - A. Right-click **My Computer** on your Desktop.
   - B. Select **Properties**.

     The **System Properties** dialog box appears.
   - C. Click the **Advanced** tab.
   - D. Click **Environment Variables**.
   - E. Under **System variables**, select the **Path** variable and click **Edit**.

     The **Edit System Variable** dialog box appears, including a text box, allowing you to edit the path variables.
   - F. Enter the `<SWIFT Install directory>/RA/lib` directory as the last entry in the **Variable value** text box. Make sure to enter a semicolon first, to separate your entry from the previous entry in the text box.
   - G. When you are finished, click **OK** to close the dialog box.
   - H. Click **OK** to close each properties dialog box.

To update the Remote API Shared Libraries environment variable for Solaris

1. Identify the installation location of your SWIFT Remote API libraries. It is usually located at `<SWIFT Install directory>/RA/lib`.

2 Update the global LD_LIBRARY_PATH environment variable to include the `<SWIFT Install directory>/RA/lib` directory as follows:

- For SH-based shells, including SH, BASH, and KSH, you must update the LD_LIBRARY_PATH variable in the **.profile** file to include the directory path, for example:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<SWIFT Install
directory>/RA/lib
```

- For CSH-based shells, including CSH and TCSH, you must update the LD_LIBRARY_PATH variable in the **.cshrc** file to include the directory path, for example:

```
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:<SWIFT Install
directory>/RA/lib
```

Note: *How this environment variable is set can vary depending on the specific shell you are using.*

To update the Remote API Shared Libraries environment variable for AIX

1 Identify the installation location of your SWIFT Remote API libraries. It is usually located at `<SWIFT Install directory>/RA/lib`.

2 Update the global LIBPATH (library path) environment variable to include the `<SWIFT Install directory>/RA/lib` directory as follows:

- For SH-based shells, including SH, BASH, and KSH, you must update the LIBPATH variable in the **.profile** file to include the directory path, for example:

```
export LIBPATH=$LIBPATH:<SWIFT Install directory>/RA/lib
```

- For CSH-based shells, including CSH and TCSH, you must update the LIBPATH variable in the **.cshrc** file to include the directory path, for example:

```
setenv LIBPATH ${LIBPATH}:<SWIFT Install directory>/RA/
lib
```

Note: *How this environment variable is set can vary depending on the specific shell you are using.*

Chapter 3

Configuring the eWay

This chapter explains how to configure the SWIFT AG eWay.

What's in This Chapter

3.1 SWIFT AG eWay Properties Dialog Box

When you install the SWIFT AG eWay, a default properties template for the eWay is also installed. You can configure the template's default properties by using the eGate Enterprise Designer. The default settings apply to all SWIFT AG eWays you use within your current Project.

You can configure the eWay using the Enterprise Designer's eWay **Properties** dialog box. This section describes general procedures on how to configure the eWay by changing the eWay's default properties.

To use the eWay Properties dialog box

- The eWay's default properties are automatically provided. You can change them, as desired.
- The **Configuration** or **Environment Configuration** folder appears already open in the left pane and displays one or more subfolders. The eWay's editable properties under the highlighted subfolder appear in the right pane.
- If there are additional subfolders, you can click any of them to display their properties.
- Many of the entries allow you to enter text. Click the desired text box, then click the ellipsis (**...**) that appears, to open a dialog box for text entry.

Note: *Even if you do not change the eWay's properties, you must open each* ***Properties*** *dialog box for each eWay and click* ***OK*** *to activate the eWay.*

For complete details on these procedures, see the *eGate Integrator User's Guide*.

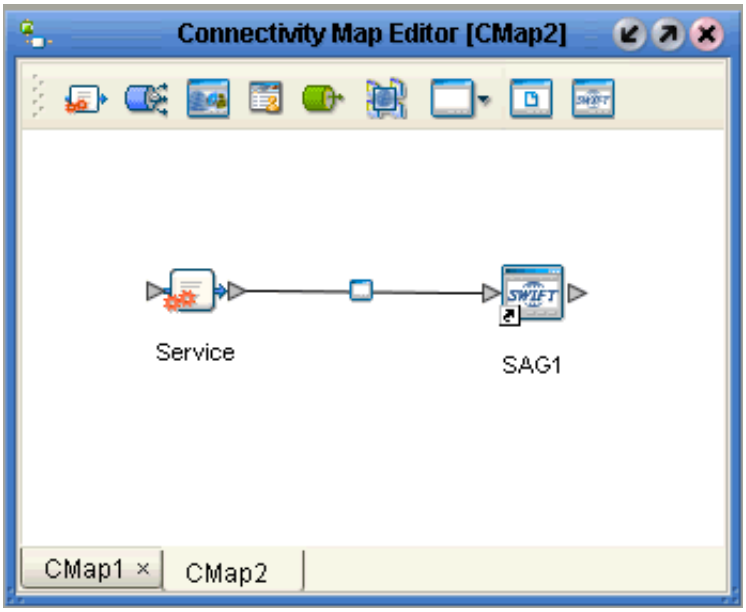
You can set properties for the eWay using either of the following Enterprise Designer user interfaces:

- Connectivity Map
- Project's Environment

To configure the SWIFT AG eWay on the Connectivity Map

1. From the eGate Enterprise Designer's **Project Explorer** create at least one Connectivity Map.
2. Create the desired external systems for your one or more Connectivity Maps. Select **SAG External Application** from the pull-down menu, for the SWIFT AG eWay.
3. Select the external application whose default eWay properties you want to change by double-clicking the **eWay** symbol. This symbol is located on the link between a **Service** and the **External Application** on the Connectivity Map canvas. See Figure 2.

Figure 2 eWay Icon



The eWay **Properties** dialog box appears. Figure 3 on page 26 shows the eWay's default properties available using the **Project Explorer** and Connectivity Map. You can use this window to modify the current eWay's properties settings.

4. Click **OK** then **Save All** to save your changes.

To configure the SWIFT AG eWay on the Project's Environment

1. On the Enterprise Designer's Connectivity Map, make sure that you have created the desired external systems (see the previous procedure).

2 Click the **Environment Explorer** tab (at the bottom of the left pane).

3 Create an Environment for your Project, then create external systems on the Environment canvas, which correspond to the systems you created using the Connectivity Map and **Project Explorer**.

4 Enter a name for each new external system.

5 Select the external system whose default eWay properties you want to change by right-clicking the desired external system's name in the **Environment Explorer** and choosing the **Properties** option from the shortcut menu.

The eWay **Properties** dialog box appears. **Figure 4 on page 29** shows the eWay's default properties available on the Project's Environment. You can use this dialog box to modify the eWay properties associated with the current external system.

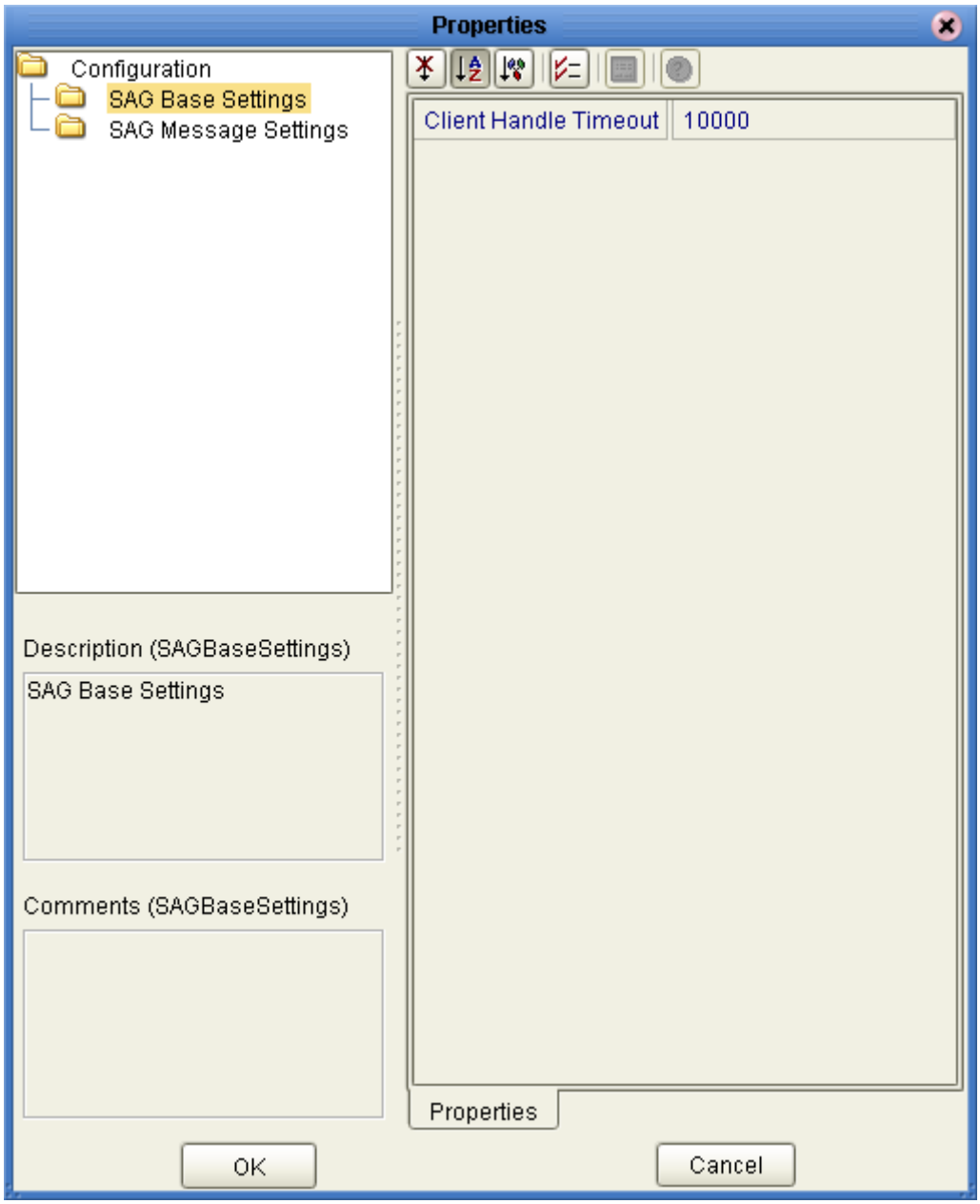
6 Click **OK** then **Save All** to save your changes.

The rest of this chapter explains the eWay's properties settings in detail, as well as how to configure them.

3.2 Configuring the eWay on the Connectivity Map

This section explains in detail how to configure the eWay's properties accessible on the Connectivity Map using the eGate Enterprise Designer. You can set these properties using the eWay **Properties** dialog box. See **Figure 3 on page 26**.

Figure 3 eWay Properties Dialog Box: Settings on the Connectivity Map



These eWay properties define the settings used to interact with the external system, under the **Configuration** folder. These properties are located in:

- **"SAG Base Settings" on page 27**
- **"SAG Message Settings" on page 27**

3.2.1 SAG Base Settings

This property allows you to define a basic connection to SWIFT Alliance Gateway made on the Connectivity Map.

Client Handle Timeout

Description

Specifies the maximum time allowed between a request message and its corresponding response message.

Required Values

The desired time in milliseconds; the default is 10,000 milliseconds or 10 seconds.

3.2.2 SAG Message Settings

These properties allow you to enter the desired default settings for SWIFT Alliance Gateway messages.

Application Id

Description

Allows you to enter the name of the SWIFT Alliance Gateway Message Partner. The Message Partner basically identifies an application that sends and receives messages on behalf of a Correspondent.

Required Values

A valid name of the desired SWIFT Alliance Gateway Message Partner; there is no default.

Context Id

Description

Allows you to enter the name of the SWIFT Alliance Gateway Context ID. This ID is used for correlating different messages. Messages with the same Context ID are conceptually related.

Required Values

A valid name of a SWIFT Alliance Gateway Context ID; there is no default.

Local Password

Description

Allows you to enter a name, called the Local Password, which is a placeholder of the signature managed by the Local Authentication Handler plug-in.

Required Values

A valid SWIFT Alliance Gateway Local Password; there is no default.

Message Format

Description

Allows you to enter the format of the current message. Each of the names of the required values denotes a SWIFT Alliance Gateway message format. You must enter the format of the current message that you are using.

Required Values

You must enter one of the following values:

- **SAG:BasicInterAct**
- **SAG:SNL**
- **SAG:Primitive**

The default is **SAG:BasicInterAct**.

Receiver

Description

Allows you to enter the name of the SWIFT Alliance Gateway Receiver. The name entered in the Receiver field is equivalent to a Correspondent as defined in the Receiver configuration of SWIFT Alliance Gateway.

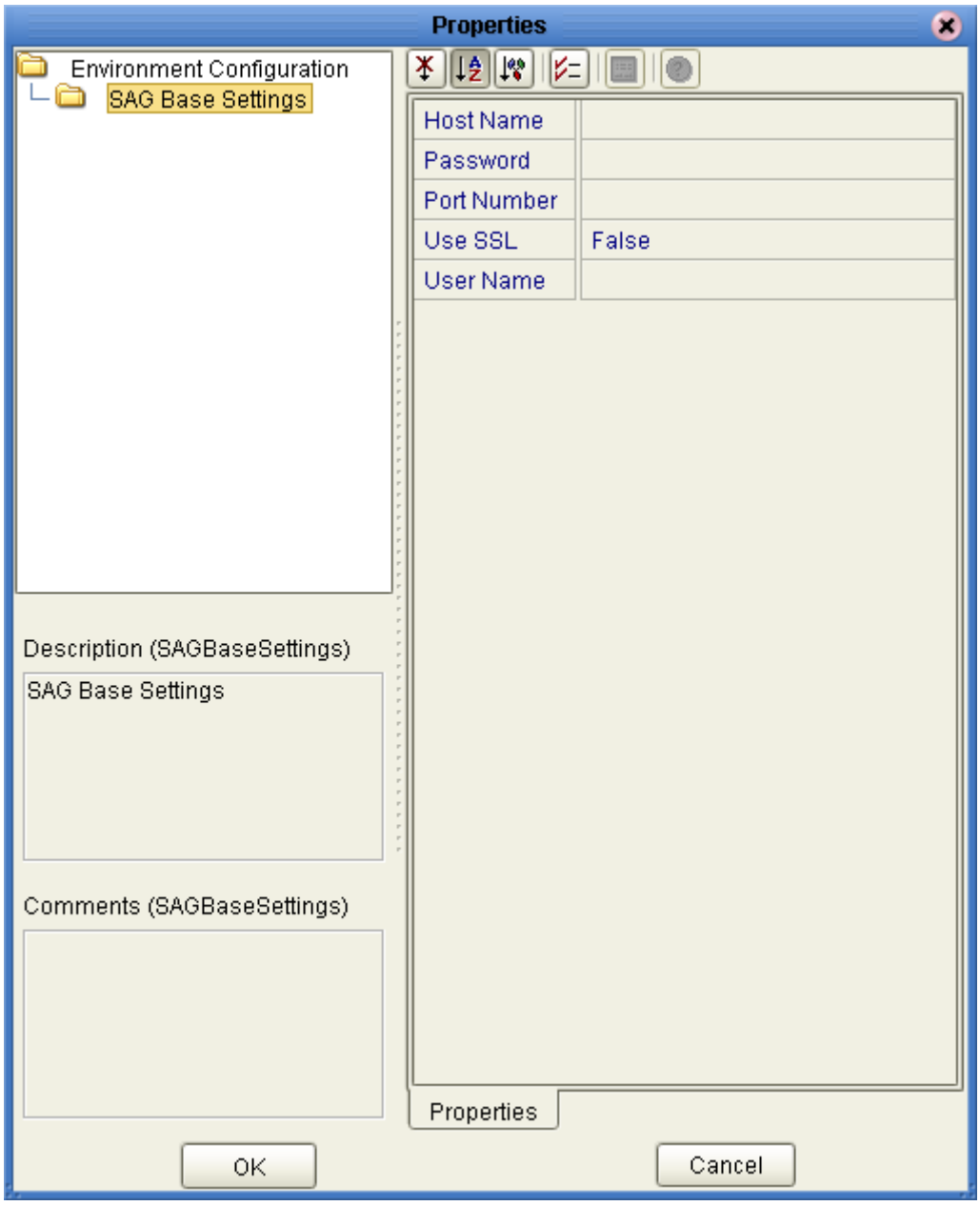
Required Values

A valid name of the desired SWIFT Alliance Gateway Receiver; there is no default.

3.3 Configuring the eWay on the Project's Environment

This section explains in detail how to configure the eWay's properties in the Project's Environment, accessible using the Enterprise Designer. You can set these properties using the eWay **Properties** dialog box. See **Figure 4 on page 29**.

Figure 4 eWay Properties Dialog Box: Settings on the Environment Explorer



These eWay properties define the basic settings used by the Project's Environment, under the **Environment Configuration** folder. These properties are located in:

- **"SAG Base Settings" on page 27**

3.3.1 SAG Base Settings

These properties allow you to define a basic connection to SWIFT Alliance Gateway made in the Project's Environment.

Host Name

Description

Allows you to enter the name of the host you want to connect to.

Required Values

A valid host name; there is no default.

Password

Description

Allows you to enter your password for connecting to the host.

Required Values

A valid user password; there is no default.

Port Number

Description

Allows you to enter the port number of the host you want to connect to.

Required Values

A valid port number; there is no default.

Use SSL

Description

Allows you to indicate whether the current connection is using the secure sockets layer (SSL) feature.

Required Values

True or **False**; the default is **False**.

User Name

Description

Allows you to enter your user name. The **User Name** setting is the same as a Correspondent as defined in the Correspondent configuration of SWIFT Alliance Gateway.

A user in SWIFT Alliance Gateway is any Correspondent sending a message. In the **SAGApplication** Object Type Definition (OTD), the Correspondent is the sender and can be overridden in a Collaboration.

In the **SAGProcessControlApplication** OTD, the Correspondent is the operator and can also be overridden in a Collaboration.

Required Values

A valid SWIFT Alliance Gateway user name, that is, the name of a Correspondent.

Chapter 4

Using the eWay OTDs

This chapter provides an overview of the SWIFT AG eWay OTDs and how to use them with the eWay and SWIFT Alliance Gateway.

What's in This Chapter

4.1 Introduction to SWIFT AG eWay OTDs

The SWIFT AG eWay has the following OTDs:

- **SAGApplication**: For application-level connectivity with and basic messaging to and from SWIFT Alliance Gateway.
- **SAGProcessControlApplication**: For administrative and command connectivity to SWIFT Alliance Gateway.

Figure 5 on page 33 shows the **SAGApplication** OTD in the eGate OTD Editor interface, the client application nodes.

Figure 5 SAGApplication in OTD Editor (Client)

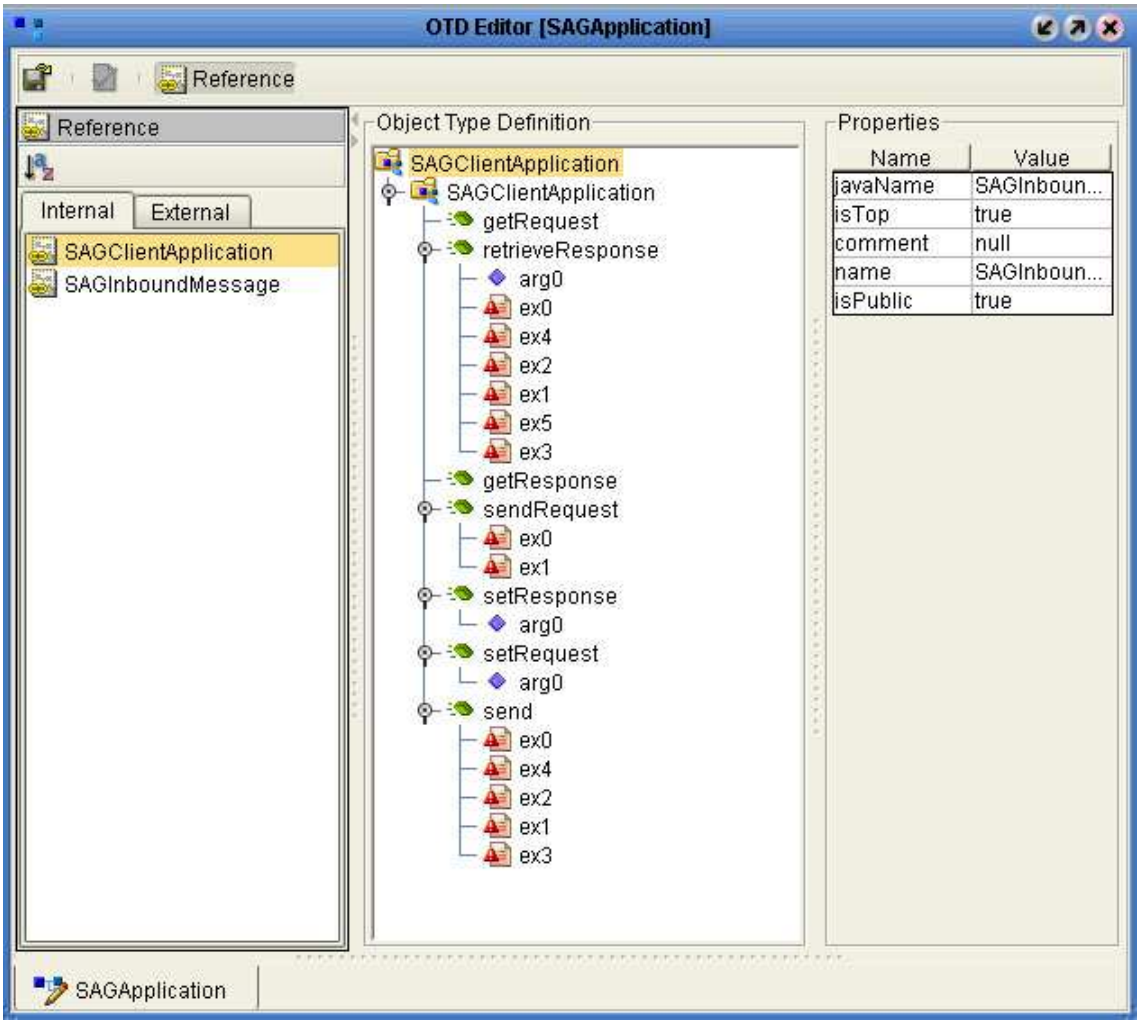


Figure 5 on page 33 shows the **SAGApplication** OTD in the eGate OTD Editor interface, the inbound message nodes.

Figure 6 SAGApplication in OTD Editor (Inbound Messages)

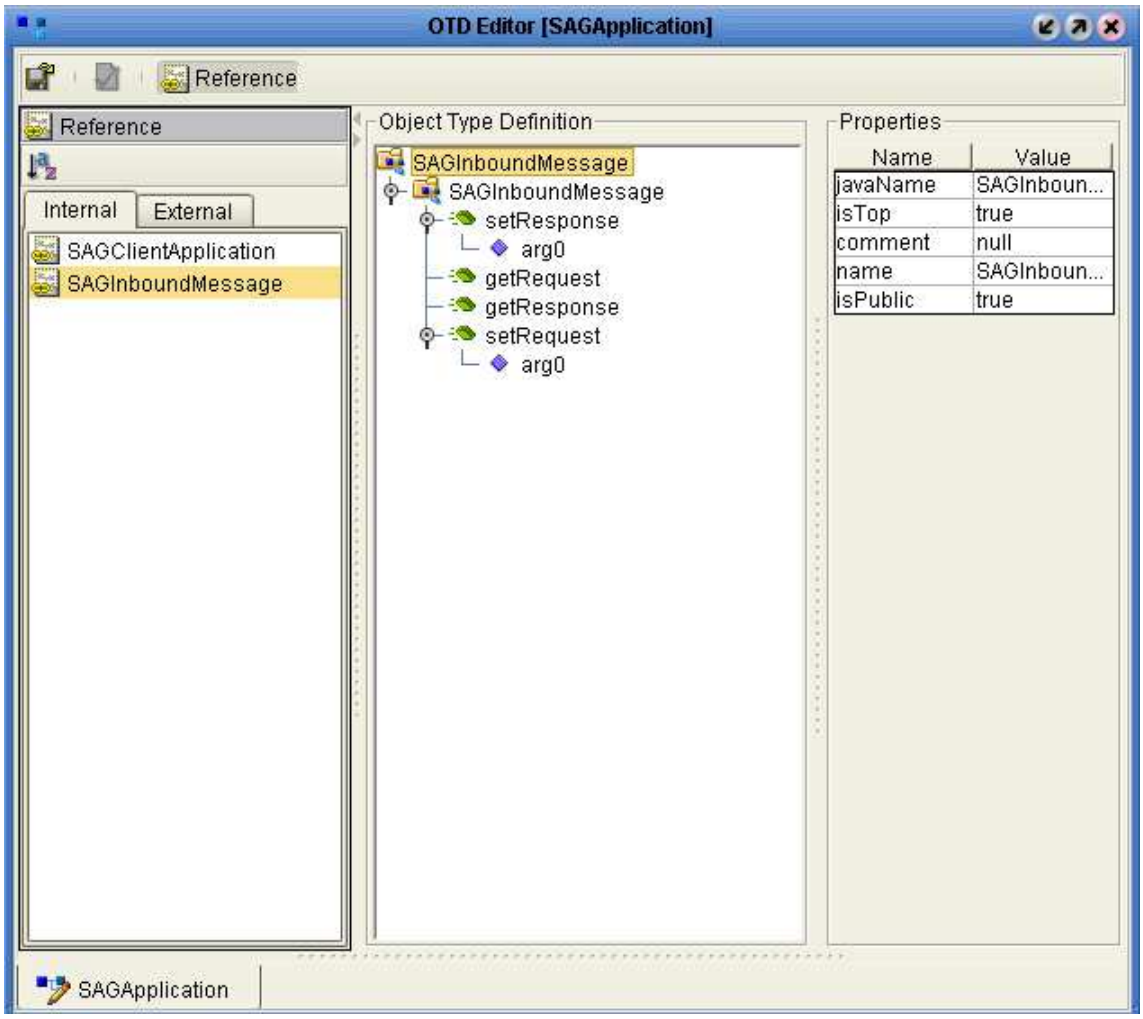
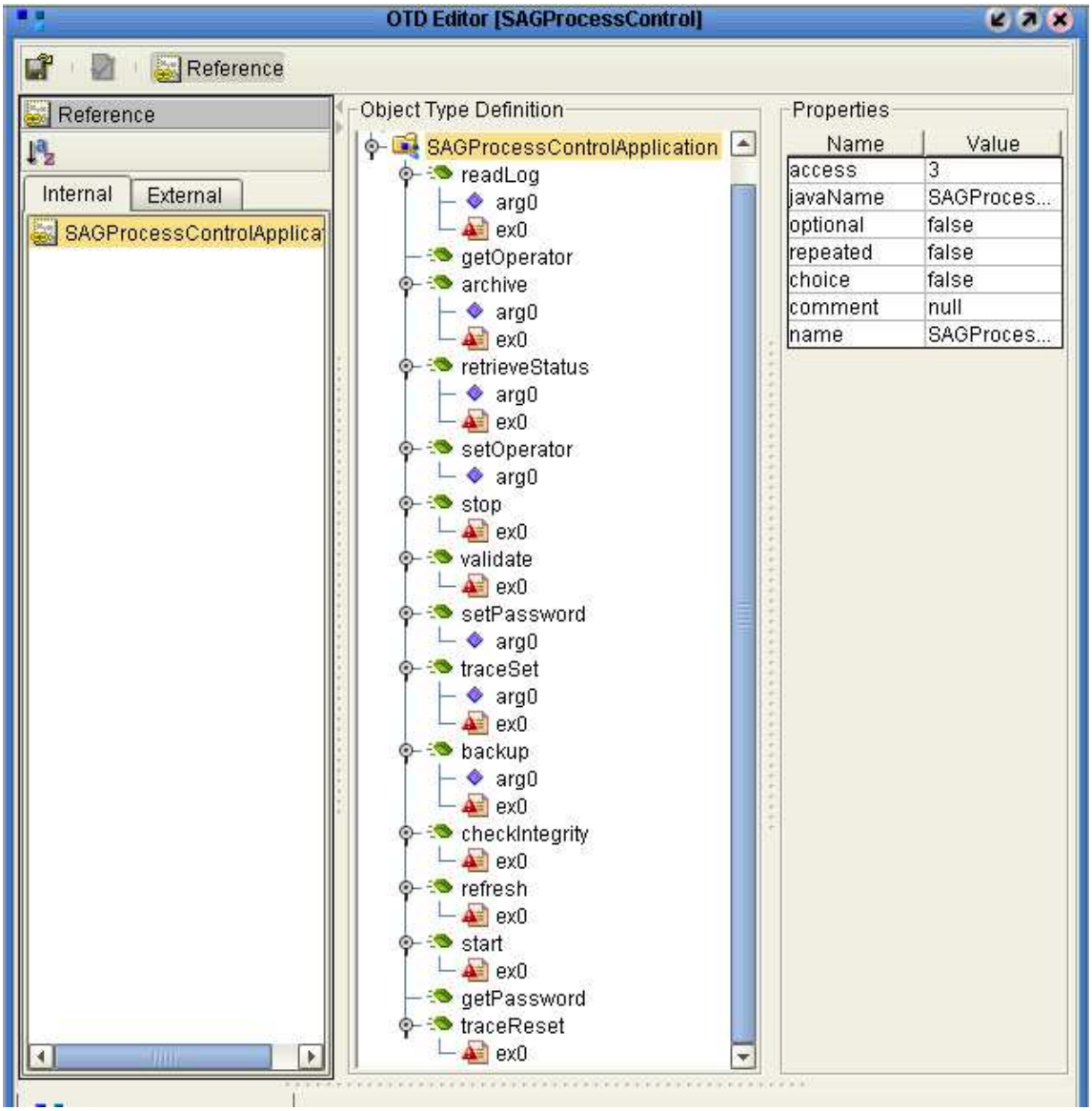


Figure 7 on page 35 shows the **SAGProcessControlApplication** OTD in the eGate OTD Editor interface.

Figure 7 SAGProcessControlApplication in OTD Editor



eWay Javadoc

See the **Javadoc** for this eWay for a complete explanation of the methods and structure of each of the classes and interfaces discussed in this chapter. See **Chapter 6** for more information on the **Javadoc** and how to access it.

The rest of this chapter explains the SWIFT AG eWay OTDs and their features in detail.

4.2 SAGApplication OTD

The **SAGApplication** OTD provides application-level connectivity to the SWIFT Alliance Gateway. Communication to and from SWIFT Alliance Gateway occurs as a set of messages. The **SAGApplication** OTD is structured to represent the communication of these messages.

See **Figure 5 on page 33** and **Figure 6 on page 34** for graphic representations of the methods of the **SAGApplication** OTD in the eGate OTD Editor interface.

4.2.1 Calling SAGApplication OTD Methods

The **SAGApplication** OTD defines the following basic methods for communicating with SWIFT Alliance Gateway:

- **send**
- **sendRequest**
- **retrieveResponse**

Communication Operation

In the eGate Collaboration Editor (Java), you can call any of the these methods. They initiate communication that operates as follows:

- The **send** method is a synchronous method for sending the request message to SWIFT Alliance Gateway. The method sends a message then blocks until a message is returned. At that point, the **SAGApplication** OTD unmarshals the contents of the message into the response message node.
- The **sendRequest** and **retrieveResponse** methods allow for asynchronous communication. The **sendRequest** method sends the request message to SWIFT Alliance Gateway, returning a token. This token can be used at a later date with the **retrieveResponse** method to get the desired response. The **retrieveResponse** method populates the response in the response message node of the OTD.

Using the SAGApplication OTD

Use the **SAGApplication** OTD as follows:

- To use the **send** method, you populate the attributes of the **Request** node, then call the **send** method. The response from SWIFT Alliance Gateway is then populated in the **Response** node.
- To use the **sendRequest** method, simply populate the attributes of the **Request** node, then all the **sendRequest** method. The response is a token that can be saved in a local variable in a Java-based Collaboration.
- To use the **retrieveResponse** method, you call the method using the token returned from the **sendRequest** method. After calling the **retrieveResponse** method, the response from SWIFT Alliance Gateway is populated by the contents of the **Response** node in the **SAGApplication** OTD.

4.2.2 SAGApplication OTD Interfaces

The **SAGApplication** OTD is represented by the following Java interface:

com.stc.connector.appconn.sag.SAGApplication

The **SAGApplication** section of the **Javadoc** explains the methods in this interface. The OTD has two subnodes. One subnode is the request message, and the other the response message. The **Request** node is represented by the following interface:

com.stc.connector.appconn.sag.SAGMessage

The **Request** node has a set of attributes that define what an acceptable SWIFT Alliance Gateway message must contain. These attributes are defined in the **SAGMessage** section of the **Javadoc**.

The **Response** node is represented by the following interface:

com.stc.connector.appconn.sag.SAGMessage

The **Response** node's attributes are completely identical to the attributes of the **Request** node. See the appropriate section of the **Javadoc** for lists of which attributes are mandatory and which are optional. Although attributes can be either mandatory or optional, all of them, except for the **Letter** attribute, can be overridden at the eWay configuration level in the Connectivity Map.

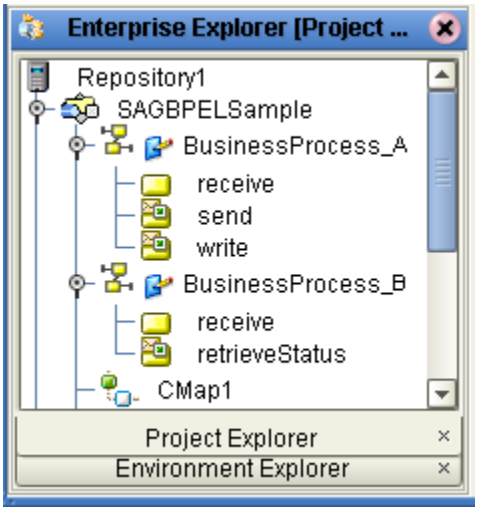
4.2.3 SAGApplication OTD BPEL Operation

With eInsight or eInsight ESB, the eWay's same basic operations are exposed as Business Process operations, using the Business Process Execution Language (BPEL). The input and output message structures of the Business Process operations are equivalent to the Java methods' structures.

For a detailed description of the **SAGApplication** OTD's BPEL Business Process operations, see the **SAGWebClientApplication** section of the **Javadoc**, which also includes the appropriate interface information

See Figure 8 for a sample list of Business Process operations as shown in the **Project Explorer**.

Figure 8 Sample Business Processes and OTDs



4.3 SAGProcessControlApplication OTD

The **SAGProcessControlApplication** OTD provides administrative connectivity to SWIFT Alliance Gateway. Its methods provide you with command-and-control features for the eWay.

See **Figure 7 on page 35** for a graphic representation of the methods of the **SAGProcessControlApplication** OTD in the eGate OTD Editor interface.

4.3.1 Calling SAGProcessControlApplication OTD Methods

The communication to the SWIFT Alliance Gateway is basically a set of commands. The OTD exposes these commands as methods.

The OTD has two attributes, **Operator** and **Password**. Both attributes are mandatory, and they can be defined at the configuration level in the eWay (external system) **Properties** dialog box on the **Environment Explorer**.

SAGProcessControlApplication OTD Method Commands

The following method commands are available through the **SAGProcessControlApplication** OTD:

- **start**
- **stop**
- **validate**

- **retrievestatus**
- **refresh**
- **traceset**
- **tracereset**
- **checkintegrity**
- **backup**
- **readlog**
- **archive**

Using the SAGProcessControlApplication OTD

Use the **SAGProcessControlApplication** OTD as follows:

- In the Collaboration Editor (Java), populate the **Operator** and **Password** attributes.

  You can also configure these attributes in the eWay **Properties** dialog box on the **Environment Explorer** and leave the attributes in the Collaboration empty.
- Call any of the methods on the **SAGProcessControlApplication** OTD (see the previous list). For detailed descriptions of each of these methods, see the **Javadoc**.

4.3.2 SAGProcessControlApplication OTD Interface

The **SAGProcessControlApplication** OTD is represented by the following Java interface:

com.stc.connector.appconn.sag.SAGProcessControlApplication

4.3.3 SAGProcessControlApplication OTD BPEL Operation

With eInsight or eInsight ESB, the eWay's same basic operations are exposed as Business Process operations, using BPEL. The input and output message structures of the Business Process operations are equivalent to the Java methods' structures.

For a detailed description of the **SAGProcessControlApplication** OTD's BPEL Business Process operations, see the **SAGWebClientApplication** section of the **Javadoc**, which also includes the appropriate interface information

See **Figure 8 on page 38** for a sample list of Business Process operations as shown in the **Project Explorer**.

Chapter 5

Reviewing the Sample Projects

This chapter describes how to implement the SWIFT AG eWay using a review of the sample Projects included with the eWay. The following Projects are included:

- Sample that employs SeeBeyond ICAN Suite's eInsight Business Process Manager and operates using eInsight Business Processes
- Sample that operates using Java-based Collaborations in eGate

This chapter assumes that you are already familiar with eGate and eInsight concepts and that you understand the basics of creating a Project using the eGate Enterprise Designer.

For a complete explanation how to use eInsight, see the *eInsight Business Process Manager User's Guide*. For a complete explanation of how to use the eGate Enterprise Designer to create and set properties for the components of an eGate Project, see the *eGate Integrator User's Guide*.

What's in This Chapter

5.1 Description of eWay Sample Projects

This section provides an overview of the eGate sample Projects for the SWIFT AG eWay and how to import and use them.

Note: *Both the BPEL and Java-based sample Projects perform the same operations. In this sense, they are identical. However, one Project operates using eInsight Business Processes, and the other using Java-based Collaborations.*

5.1.1 Projects and the Enterprise Designer

A Project contains all of the eGate components you designate to perform one or more desired processes in eGate. Each eGate Project is created using the Enterprise Designer.

The Project canvas contains windows that represent the various stages of your Project. The types of windows in your Project canvas area include:

- **Connectivity Map Canvas**: Contains the eGate business logic components, such as Collaborations, Topics, Queues, and eWays, that you include and configure in the structure of the Project.
- **OTD Editor**: Contains the source files used to create Object Type Definitions (OTDs) to use with a Project.
- **Business Process Canvas**: Allows you to use eInsight's Business Process management features that use a Business Process Execution Language (BPEL) interface; you can use this canvas in eInsight ESB or in eGate with eInsight.
- **Collaboration Editor (Java)**: Allows you to create and/or modify Business Rules to implement the business logic of a Project's Java-based Collaboration Definitions in eGate.

5.1.2 Importing Sample Projects

Before you can view or work with a sample Project, you must first import it into eGate, using the Enterprise Designer.

Note: *The sample .**zip** file you first download may contain more than one Project and/or additional files. If this is the case, you must unzip this file first, find the desired Project file, then import the Project file. For information on how to obtain this file, see* **Chapter 2**. *For complete instructions on downloading and installing the eWay and related files, see the **SeeBeyond ICAN Suite Installation Guide**.*

The container file you are looking for is **SWIFT_AG_eWay_Sample.zip**. The Project file names are:

- **SWIFT_AG_SampleProject_BPEL.zip**: BPEL Project
- **SWIFT_AG_SampleProject_JCE.zip**: Java-based Collaboration Project

You can name imported Projects as desired.

To import a sample Project

1. Save any changes not saved previously.
2. From the Enterprise Designer's **Project Explorer** pane, right-click the desired Repository and select **Import**.
3. On the **Import Manager** window, browse to the directory that contains the sample Project **.zip** file.
4. Select the sample Project file and click **Open**.

5 Click **Import**. If the import was successful, click **OK** on the **Import Status** dialog box.

6 Close the **Import Manager** window.

Important: *An imported Project does not contain an environment or a deployment profile. After importing a Project, you must use the Enterprise Designer to create these functions for the Project. See* **"Creating the Project's Environment" on page 53** *and* **"Deploying a Project" on page 54***. For additional information, see the* ***eGate Integrator User's Guide*** *and* ***SeeBeyond ICAN Suite Deployment Guide****.*

You must check out the major eGate components before you can change them. For details, see the *eGate Integrator User's Guide*.

5.1.3 Configuring SWIFT Alliance Gateway for Sample Projects

The sample Projects demonstrate a connection between two SWIFT Alliance Gateway Correspondents. To use the sample Projects, you must configure the SWIFT Alliance Gateway to route messages between the Correspondents properly.

Creating the Message Partners

In the **MessagePartner** Module Panel of the SWIFT Alliance Gateway system administrator user interface, you must create new Message Partners. These Message Partners perform the work on behalf of the SWIFT Alliance Gateway client (in these procedures called the **SAGClient**) and the server (called the **SAGServer**).

To create the new Message Partners in the MessagePartner Module Panel

1 To create a new Message Partner for the **SAGClient**, enter the following values for each field:

- **Name: SAGClientMP**
- **Type: Client**
- **Default Message Format: Basic InterAct**

2 To create a new Message Partner for the **SAGServer**, enter the following values for each field:

- **Name: SAGServerMP**
- **Type: Server**
- **Host Adapter: RA Host Adapter**

3 Make the following entries under **Supported Message Formats**:

- **Primitive Format**
- **SNL Format**
- **Basic Interact Format**

Creating the Correspondents

In the **Correspondent** Module Panel of the SWIFT Alliance Gateway system administrator user interface, you must create new Correspondents. These Correspondents act as the **SAGClient** and the **SAGServer**.

To create the new Correspondents in the Correspondent Module Panel

1. To create a new Correspondent for the **SAGClient**, enter the following values for each field:
   - **Name: SAGClient**
   - **Type: Application**
   - **Nature: External**
   - **Default Interface: Application Interface**
   - **Plug-in Configuration: Application Interface**

   An **Application Interface** tab appears when you make the entry for **Plug-in Configuration**.
2. Click the (new) **Application Interface** tab.

   A field appears for a single entry, for the **MessagePartner**.
3. Enter **SAGClientMP**.
4. To create a new **Correspondent** for the **SAGServer**, enter the following values for each field:
   - **Name: SAGServer**
   - **Type: Application**
   - **Nature: External**
   - **Default Interface: Application Interface**
   - **Plug-in Configuration: Application Interface**

   An **Application Interface** tab appears when you make the entry for **Plug-in Configuration**.
5. Click the (new) **Application Interface** tab.

   A field appears for a single entry, for the **MessagePartner**.
6. Enter **SAGServerMP**.

Setting up Routing Between Message Partners

In the **Endpoints** Module Panel of the SWIFT Alliance Gateway system administrator user interface, you must create new Endpoints. These Endpoints route messages to and from the **SAGClient** and **SAGServer**.

If you use Endpoints for message routing between Message Partners, you must create two. One Endpoint sends messages from the **SAGClient** to the **SAGServer**. The other allows the **SAGServer** to reply.

Note: *There are a variety of different ways to route messages through SWIFT Alliance Gateway. The Project samples employ EndPoints, using this module for message routing. However, other routing methods are possible.*

To create new Endpoints in the Endpoint Module Panel

1. To create a new Endpoint to route messages from the **SAGClient** to the **SAGServer**, enter the following values for each field:
   - **Name: SAGClientToSAGServer**
   - **From: Application Interface**
   - **From Data: SAGClientMP**
   - **To: Application Interface**
   - **To Data: SAGServerMP**
2. To create a another Endpoint to route messages from the **SAGServer** to the **SAGClient**, enter the following values for each field:
   - **Name: SAGServerToSAGClient**
   - **From: Application Interface**
   - **From Data: SAGServerMP**
   - **To: Application Interface**
   - **To Data: SAGClientMP**

5.1.4 Basic eWay Components

The eWay's sample Projects allow you to see how to set up the eWay to perform operations within eGate. To use the SWIFT AG eWay in Projects, you must also set, that is, configure the desired eWay properties.

SWIFT AG eWay Properties

The properties for the SWIFT AG eWay contain the settings used to connect with a specific external system. You can configure these settings using the eGate eWay **Properties** dialog box. For more information about SWIFT AG eWay properties and this dialog box, see **Chapter 3**.

eWay OTDs

Two versatile OTDs come packaged with the eWay. These OTDs contain eWay properties and methods and also allow you to utilize, as well as extend, the eWay's features. See **Chapter 4** and the **Javadoc** for details.

5.2 Overview: Sample Project Using BPEL

This section explains generally how to implement the SWIFT AG eWay using the eInsight Project sample that allows you to use a BPEL interface. This sample is included on your installation CD-ROM. You can use this sample with eInsight ESB or eGate with eInsight.

The extracted sample Project file is named **SWIFT_AG_SampleProject_BPEL.zip**. This Project allows you to observe an end-to-end data-exchange scenario involving eGate and the SWIFT AG eWay.

For instructions on how to import the sample Project, see the **procedure on page 41**. For an overview of the sample Project and what is does, see **"Overview: Sample Project Using BPEL" on page 45**.

See the *Insight Business Process Manager User's Guide* for details on how to build an end-to-end Project using the eInsight BPEL interface.

5.2.1 Using the eWay With eInsight

You can set up and deploy an eGate component using eInsight or using eInsight ESB only. Once you have associated the desired component with a Business Process, the eInsight engine can automatically invoke that component during run time, using the eInsight BPEL interface.

Note: *You must have the **eInsight.sar** or **eInsightESB.sar** file installed to use the features explained in this section. See the **SeeBeyond ICAN Suite Installation Guide** for complete installation procedures.*

Using eInsight With eGate Components

eInsight operates seamlessly with eGate. Examples of eGate components that can interface with eInsight using Business Processes are:

- Java Messaging Service (JMS)
- Object Type Definitions (OTDs)
- An eWay
- eGate Services

Using the eGate Enterprise Designer and its eInsight canvas, you can add a desired operation to a Business Process, then associate that process with an eGate component, for example, a Service. In the Enterprise Designer, you can associate the Business Process and Service icons using drag-and-drop operations.

See the *eInsight Business Process Manager User's Guide* for details.

SWIFT AG eWay With eInsight

You can add SWIFT AG eWay objects to an eInsight Business Process during the system design phase. To make this association, select the desired operation, for example **send**, **sendRequest**, or **retrieveResponse,** under the eWay in the Enterprise Explorer and drag it onto the eInsight Business Process canvas. In turn, you can activate a Business Process in eGate by dragging it onto a Service or onto the Business Process canvas.

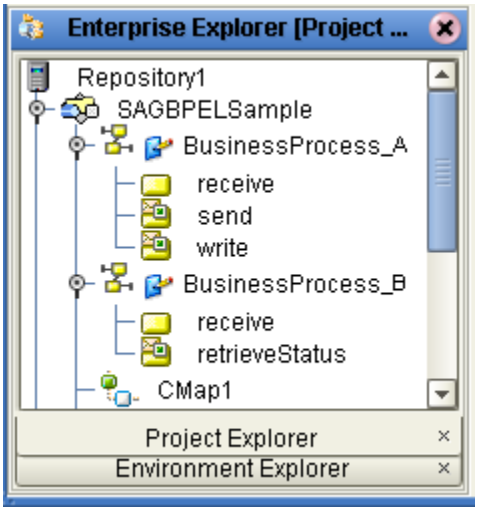
At run time, the eInsight Engine is able to invoke each of the operations in order as set up in the Business Process. Using the engine's BPEL interface, eInsight in turn invokes the SWIFT AG eWay operations, as well as those of any other eWays in the current Project.

The operations contained in the BPEL interface are available using the SWIFT AG eWay's OTDs, that is:

- **SAGApplication**
- **SAGProcessControlApplication**

See Figure 9 for a sample list of Business Process operations as shown in the **Project Explorer**.

Figure 9 Sample Business Processes and OTDs



These operations are the same as the methods used by the OTDs. For more information on the OTDs and how to use them, see **Chapter 4** and the **Javadoc**.

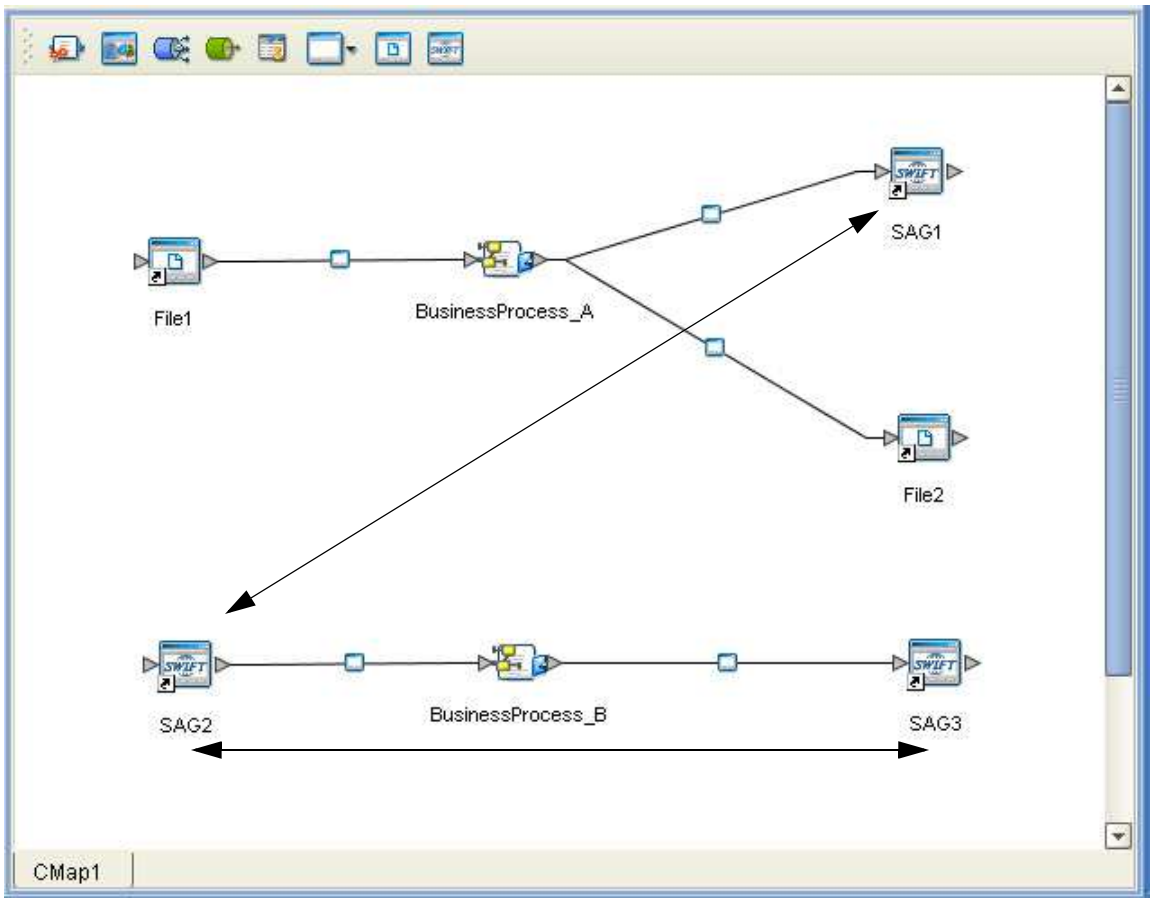
5.2.2 BPEL Sample Project Summary

The sample BPEL Project demonstrates how the SWIFT AG eWay processes information from a SWIFT system through Business Processes you set up using the eInsight BPEL interface.

The sample Project demonstrates how the **SAGApplication** and **SAGProcessControlApplication** OTDs work in a BPEL data-processing scenario.

Figure 10 shows a general diagram of how this scenario operates, using an eGate Connectivity Map.

Figure 10 Sample eInsight/BPEL Business Process Project Scenario



Project Components

The Project has the following components:

- File external application (inbound): **File1**
- Inbound File eWay
- Business logic implementation employs an eInsight Business Process (eGate Service component) for processing data: **BusinessProcess_A**
- SWIFT AG eWay (for **SAG1**)
- External SWIFT Alliance Gateway system for **BusinessProcess_A**: **SAG1**
- Outbound File eWay
- File external application (outbound): **File2**
- SWIFT AG eWay (for **SAG2**)
- External SWIFT Alliance Gateway system (inbound): **SAG2**
- Additional eInsight Business Process for processing data: **BusinessProcess_B**
- SWIFT AG eWay (for **SAG3**)
- External SWIFT Alliance Gateway system (outbound): **SAG3**

Project Operation

The BPEL Project operates as follows:

- The object of the Project's operation is to query the system status of the **SAG3** SWIFT Alliance Gateway application through the specific arrangement of components set up in the sample Project. The query starts from an external file.
- The **File** OTD in the File eWay reads an input data file from an external file system (**File1**). The file contains the detail level of the status you want. This level may be one of the following strings:
  - **System**
  - **PlugIn**
  - **Executable**
- The input data in the sample asks for **System** (system status) and starts the data exchange within **BusinessProcess_A** that interacts between the **File** OTD and the SWIFT AG eWay's **SAGApplication** OTD.
- A **SAGApplication** OTD, acting as a client, takes the string (**System**) and sends it as a **Basic:InterAct** message to another **SAGApplication** OTD acting as a server for the **SAG2** external application.
- The server **SAGApplication** OTD sends the string (**System**) to **BusinessProcess_B**.
- **BusinessProcess_B**, invokes the **retrieveStatus** operation on the **SAGProcessControlApplication** OTD, thereby communicating the requested status level to **SAG3** external application.

- The resulting system status (**System**) from **SAG3** is passed back to the server **SAGApplication** OTD in **BusinessProcess_B**, which in turn gets passed back to the client **SAGApplication** OTD in **SAG2**.
- The query result is returned from **SAG2** to **SAG1**, then passed on to the File eWay through **BusinessProcess_A**.
- The status result is written (outbound) to a file external application (**File2**).

The final goal of the project is to query the **SAG3** SWIFT Alliance Gateway installation for its status. It is possible to call the **SAGProcessControlApplication** OTD directly. However, the point of the Project is to demonstrate all aspects of the the eWay's OTDs.

Input and Output Data

When you extract the sample Project file, **SWIFT_AG_eWay_Sample.zip**, you get the following input and output data files to be used with this (BPEL) Project:

- **inputsagBPEL.~in**
- **outputsagBPEL1.dat**

5.3 Overview: Sample Collaboration (Java) Project

This section explains generally how to implement the SWIFT AG eWay using the eGate Project sample that includes a Java-based Collaboration. This sample is included on your installation CD-ROM.

The extracted sample Project file is named **SWIFT_AG_SampleProject_JCE.zip**. This Project allows you to observe an end-to-end data-exchange scenario involving eGate and the SWIFT AG eWay.

For instructions on how to import the sample Project, see the **procedure on page 41**. For an overview of the sample Project and what is does, see **"Overview: Sample Collaboration (Java) Project" on page 49**.

See the *eGate Integrator User's Guide* for details on how to build an end-to-end Project in eGate.

5.3.1 Using the eWay With Java-based Collaborations

Java-based Collaborations in eGate use the SWIFT AG eWay OTDs to process SWIFT data within the eGate system. Using Collaboration Definitions and these OTDs, you can allow the eWay to access the desired SWIFT messaging functions.

The SWIFT AG eWay's OTDs are:

- **SAGApplication**
- **SAGProcessControlApplication**

For more information on these OTDs and how to use them, see **Chapter 4**.

Creating Business Rules Within Collaboration Definitions

A Collaboration Editor (Java) allows you to create the Business Rules that implement your business logic for a Java-based Collaboration Definition.

See the *eGate Integrator User's Guide* for complete information on how to use the Collaboration Editor (Java).

Collaboration Editor (Java) Window

The Collaboration Editor (Java) window displays in the Enterprise Designer after you create a Java-based Collaboration Definition. You can also open this editor by right-clicking on the name of the desired Collaboration Definition in the **Project Explorer** and choosing **Open** from the shortcut menu.

To complete a Java-based Collaboration Definition, you can use this editor to create Business Rules.

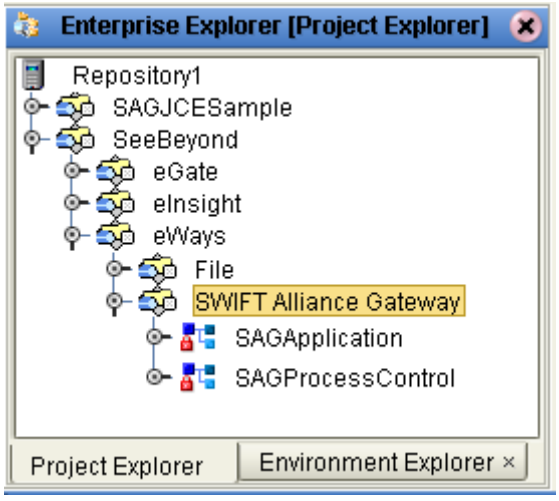
You can create the desired Business Rules for your Project by dragging and dropping values from a source SWIFT AG eWay OTD onto the nodes of a destination OTD and/or other OTDs. The OTD nodes represent SWIFT Alliance Gateway messaging and control functions, which are in turn able to call the desired methods.

5.3.2 Collaboration Definitions and OTDs in Sample

The sample comes with Java Collaboration Definitions already built for you. These Collaboration Definitions model the basic SWIFT Alliance Gateway operations enabled by the SAG eWay's OTDs.

Figure 11 shows the OTDs available to the Java-based Collaboration Definitions, as shown in the **Project Explorer**.

Figure 11 OTDs for Java-based Collaborations



For more information on how to use these OTDs with Java-based Collaborations, see **Chapter 4** and the **Javadoc**.

5.3.3 Java-based Sample Project Summary

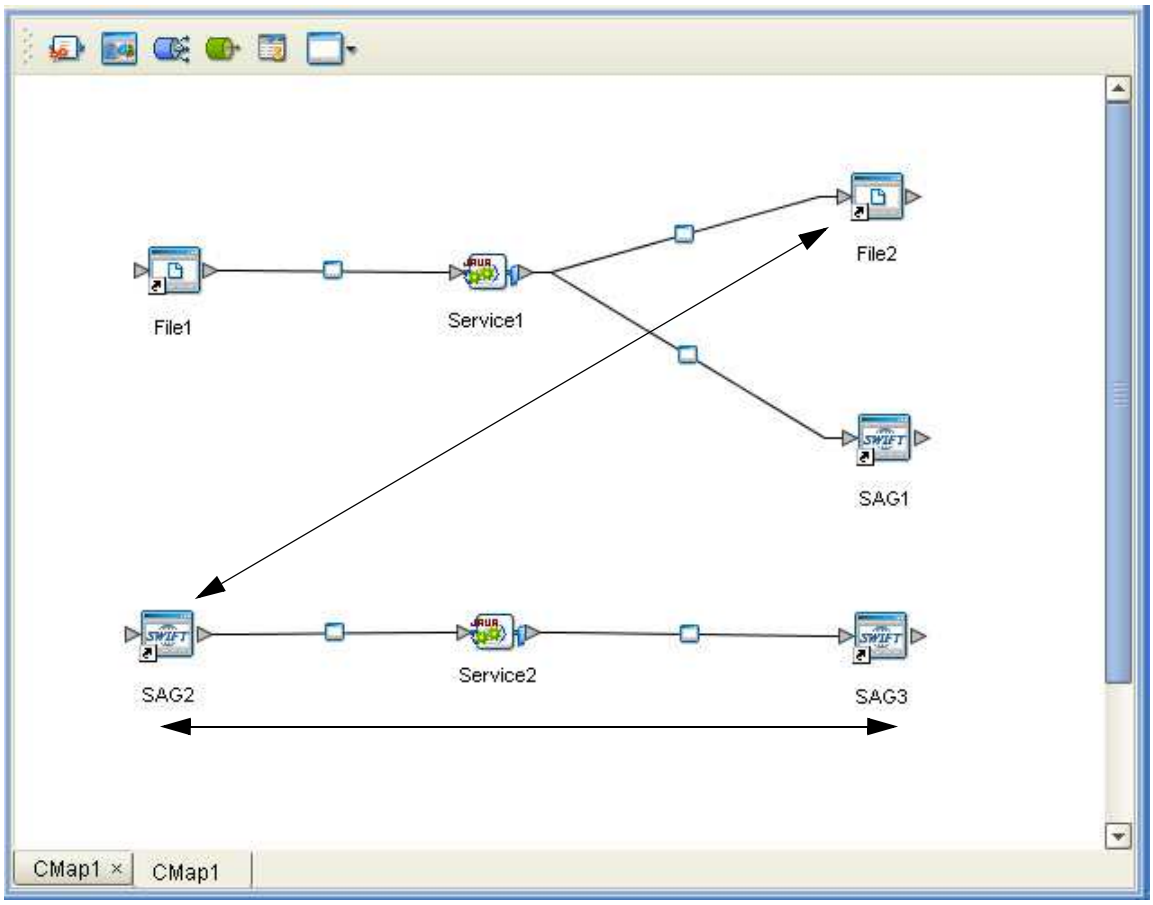
The sample Java-based Project demonstrates how the SWIFT AG eWay processes information from a SWIFT system through Business Rules you set up using Java-based Collaborations in eGate.

Note: *The operation of this sample Project is almost identical to that of the BPEL sample Project, except that the current Project uses Java-based Collaborations to process the data instead of eInsight Business Processes.*

The sample Project demonstrates how the **SAGApplication** and **SAGProcessControlApplication** OTDs work in a Java-based Collaboration data-processing scenario.

Figure 12 shows a general diagram of how this scenario operates, using an eGate Connectivity Map.

Figure 12 Sample Java-based Collaboration Project Scenario



Project Components

The Project has the following components:

- File external application (inbound): **File1**
- Inbound File eWay
- Business logic implementation employs an eInsight Business Process (eGate Service component) for processing data: **Service1**
- SWIFT AG eWay (for **SAG1**)
- External SWIFT Alliance Gateway system **SAG1** (for **Service1**)
- Outbound File eWay
- File external application (outbound): **File2**
- SWIFT AG eWay (for **SAG2**)
- External SWIFT Alliance Gateway system (inbound): **SAG2**
- Additional eInsight Business Process for processing data: **Service2**
- SWIFT AG eWay (for **SAG3**)
- External SWIFT Alliance Gateway system (outbound): **SAG3**

Project Operation

The Java-based Project operates as follows:

- The object of the Project's operation is to query the system status of the **SAG3** SWIFT Alliance Gateway application through the specific arrangement of components set up in the sample Project. The query starts from an external file.
- The **File** OTD in the File eWay reads an input data file from an external file system (**File1**). The file contains the detail level of the status you want. This level may be one of the following strings:
  - **System**
  - **PlugIn**
  - **Executable**
- The input data in the sample asks for **System** (system status) and starts the data exchange within **Service1** that interacts between the **File** OTD and the SWIFT AG eWay's **SAGApplication** OTD.
- A **SAGApplication** OTD, acting as a client, takes the string (**System**) and sends it as a **Basic:InterAct** message to another **SAGApplication** OTD acting as a server for the **SAG2** external application.
- The server **SAGApplication** OTD sends the string (**System**) to **Service2**.
- **Service2**, invokes the **retrieveStatus** method on the **SAGProcessControlApplication** OTD, thereby communicating the requested status level to **SAG3** external application.

- The resulting system status (**System**) from **SAG3** is passed back to the server **SAGApplication** OTD in **Service2**, which in turn gets passed back to the client **SAGApplication** OTD in **SAG2**.
- The query result is returned from **SAG2** to **SAG1**, then passed on to the File eWay through **Service1**.
- The status result is written (outbound) to a file external application (**File2**).

The final goal of the project is to query the **SAG3** SWIFT Alliance Gateway installation for its status. It is possible to call the **SAGProcessControlApplication** OTD directly. However, the point of the Project is to demonstrate all aspects of the the eWay's OTDs.

Note: *Keep in mind that **Service1** is called **Collaboration_1** in the **Project Explorer** and **Service2** is called **Collaboration_2**.*

Input and Output Data

When you extract the sample Project file, **SWIFT_AG_eWay_Sample.zip**, you get the following input and output data files to be used with this (Java-based) Project:

- **inputsagJCE.~in**
- **outputsagJCE1.dat**

5.4 Creating the Project's Environment

This section provides general procedures for creating an Environment for your Project. For a complete explanation, see the *eGate Integrator User's Guide.*

To create an Environment

1. From the Enterprise Designer, click the **Environment Explorer** tab on the Enterprise Explorer.
2. Under the current **Repository** icon in the **Environment Explorer**, create a new Environment for your Project and name it as desired.
3. In the **Environment Explorer**, right-click the **Environment** icon and select the desired external systems from the shortcut menu. Include external systems for the SWIFT AG eWay and the File eWays (inbound and outbound). Give them the same names as you did the corresponding external applications on the Connectivity Map.
4. Use the same shortcut menu to create a Logical Host for your Project, and name it as desired.
5. Click **Save** and return to the **Project Explorer** tab.

5.5 Setting eWay Properties

You must set the eWay properties for your specific system and for the current Project, using the eGate Enterprise Designer. For directions on accessing and using the eWay **Properties** dialog box, as well as a complete explanation of the SWIFT AG eWay properties, see **Chapter 3**.

To set properties for the File eWays

1. From the **Project Explorer**, open the **CMap** Connectivity Map for the sample Project.
2. To change the default properties for the inbound File eWay, double-click the **FileIn** external system's eWay icon. For this eWay, select the **Inbound** properties.

   The eWay **Properties** dialog box appears. This eWay can use the current default properties settings for the sample Project. However, if you are using different file names and/or folders than the defaults, you must enter the names of the files/folders you are using.

Note: *Even if you do not change the eWay's properties, you must open each **Properties** dialog box for every eWay and click **OK** to activate the eWay.*

3. For this sample, use the **C:\temp** as the property for **Directory**, and for **Input file name**, enter ***.txt**.
4. Click **OK** to save the settings and close the eWay **Properties** dialog box.
5. To change the default properties for the outbound File eWay, double-click the **FileOut** external system's eWay icon. For this eWay, select the **Outbound** properties.

   Use your file and folder names if they differ from the defaults. For all other properties, you can use the defaults.
6. Click **OK** to close the properties dialog box and save your changes.

To set properties for the SWIFT AG eWays

1. To begin changing the default properties for the SWIFT AG eWay, double-click the SWIFT AG eWay external system's eWay icon.

   The eWay **Properties** dialog box appears.
2. The properties settings appear in the **Properties** pane on the left.
3. Set the properties as desired then click **OK** to close the dialog box and save.

   See **Chapter 3** for details on how to set SWIFT AG eWay properties.

5.6 Deploying a Project

This section provides general procedures for Project deployment.

5.6.1 Basic Steps

For a complete explanation of how to deploy and run an eGate Project, see the *eGate Integrator User's Guide.*

To deploy the Project

1. From the **Project Explorer**, select the current Project and right-click, choosing **New > Deployment Profile** from the shortcut menus.
2. From the **Create a Deployment Profile** dialog box, enter the name of the current Project and select the Environment you created for this Project.
3. Click **OK**.

   The Deployment Profile canvas appears as follows:

   - The Project's external applications and Services show up as items on the left side of the canvas.
   - The external systems and Logical Host you created under **"Creating the Project's Environment" on page 53** show up as windows on the right side of the canvas.
4. Set up your Deployment Profile by dragging the items on the left into the corresponding window on the right.
5. Click **Save All** then **Activate**.

   When the Project has been activated, a pop-up message appears stating the activation was successful.

For additional information, see the *SeeBeyond ICAN Suite Deployment Guide.*

To run the Project

For instructions on how to run a Project, see the *eGate Integrator Tutorial*.

5.6.2 Alerting and Logging

eGate provides an alerting and logging feature. This feature allows monitoring of messages and captures any adverse messages in the order of severity, based on a configured severity level and higher. For details on how to enable the Logging feature, see the *eGate Integrator User's Guide*.

Chapter 6

Using eWay Java Methods

This chapter provides an overview of the Java classes and methods contained in the SWIFT AG eWay. These methods are used to extend the functionality of the eWay.

What's in This Chapter

6.1 SWIFT AG eWay Methods: Overview

The SWIFT AG eWay exposes various Java methods to add extra functionality to the eWay. These methods make it easier to set information in the SWIFT AG eWay Object Type Definitions (OTDs), as well as get information from them.

6.1.1 Relation to eWay Properties

The nature of this data transfer depends on the properties you configure and set for the eWay in the eGate Enterprise Designer. For more information on the eWay's properties, see **Chapter 3**.

6.1.2 SWIFT AG eWay Javadoc

For a complete list of the Java methods within the classes listed in this chapter, refer to the **Javadoc**. You can download the **Javadoc** using the Enterprise Manager, while you are installing the eWay.

The download file is named **SWIFT_AG_eWay_Javadoc.zip**. For more information on how to download this file, see **Chapter 2**.

For complete instructions on downloading and installing the eWay and related files, see the *SeeBeyond ICAN Suite Installation Guide*.

6.2 eWay Java Classes and Interfaces

You can see the **JavadocOverview.html** file for a list and summary of all the eWay's Java classes/interfaces. The SWIFT AG eWay provides the following Java classes and interfaces:

- **SAGApplicationException**
- **SAGApplicationTimeoutException**
- **SAGArchiveSettings**
- **SAGBackupSettings**
- **SAGClientApplication**
- **SAGCredentials**
- **SAGIdentifier**
- **SAGInboundMessage**
- **SAGLogSettings**
- **SAGLostConnectionException**
- **SAGLostMessageException**
- **SAGMessage**
- **SAGProcessControlApplication**
- **SAGProcessControlWebApplication** (class and interface)
- **SAGResult**
- **SAGStatusMessage**
- **SAGTraceSettings**
- **SAGUnknownTokenException**
- **SAGUnregisteredMessagePartnerException**
- **SAGWebClientApplication**

Index

R

S