SeeBeyond ICAN Suite

# SWIFT ADK eWay Intelligent Adapter User's Guide

Release 5.0



SeeBeyond Proprietary and Confidential

The information contained in this document is subject to change and is updated periodically to reflect changes to the applicable software. Although every effort has been made to ensure the accuracy of this document, SeeBeyond Technology Corporation (SeeBeyond) assumes no responsibility for any errors that may appear herein. The software described in this document is furnished under a License Agreement and may be used or copied only in accordance with the terms of such License Agreement. Printing, copying, or reproducing this document in any fashion is prohibited except in accordance with the License Agreement. The contents of this document are designated as being confidential and proprietary; are considered to be trade secrets of SeeBeyond; and may be used only in accordance with the License Agreement, as protected and enforceable by law. SeeBeyond assumes no responsibility for the use or reliability of its software on platforms that are not supported by SeeBeyond.

SeeBeyond, e\*Gate, e\*Way, and e\*Xchange are the registered trademarks of SeeBeyond Technology Corporation in the United States and/or select foreign countries. The SeeBeyond logo, SeeBeyond Integrated Composite Application Network Suite, eGate, eWay, eInsight, eVision, eXchange, eView, eIndex, eTL, ePortal, eBAM, and e\*Insight are trademarks of SeeBeyond Technology Corporation. The absence of a trademark from this list does not constitute a waiver of SeeBeyond Technology Corporation's intellectual property rights concerning that trademark. This document may contain references to other company, brand, and product names. These company, brand, and product names are used herein for identification purposes only and may be the trademarks of their respective owners.

© 2004 by SeeBeyond Technology Corporation. All Rights Reserved. This work is protected as an unpublished work under the copyright laws.

This work is confidential and proprietary information of SeeBeyond and must be maintained in strict confidence. Version 20040806171856.

# Contents

Chapter 1	
Introducing the SWIFT ADK eWay	6
SWIFT Overview	6
SWIFT ADK eWay Overview Types of SWIFT ADK eWay Message Flows Publish (outbound asynchronous) Request / Reply (outbound synchronous) Subscribe (inbound asynchronous)	7 8 8 9 10
Chapter 2	
Installing the SWIFT ADK eWay	12
Supported Operating Systems	12
System Requirements	12
External System Requirements	13
Before Installing the eWay	14
eWay Installation Procedures	14
After Installation	15
Chapter 3	
Configuring the SWIFT ADK eWay	16
SWIFT ADK eWay Configuration	16
Configuring the SeeBeyond JMS IQ Manager	17
Enable authentication and authorization	18 18
Host Name	18
Server Port Server SSI Port	18 10
	17

### **Chapter 4**

Installing and Setting up the SEWS eWay Component	20
Overview	20
Installing the SEWS Component on Windows	21
Installing the SEWS Component on UNIX	23
Configuring SEWS System Management Window Network Parameters IP Address or host name Listening TQ Name Message Server Login Message Type Sending TQ Name TCP Port Number Use SSL Flag Security Definition Window Secret	26 26 27 27 29 30 31 32 33 33 33
Starting the SEWS Swift ADK eWay Component	35
Setting Up a Test Environment (Optional) Queues Routing Points	36 36 36
Authentication	37

### **Chapter 5**

Using JMS OTD with the SEWS Component	38
Publish to SEWS Asynchronously Setting the JMS Header property as "PUT" Setting the JMS Header property as "LIST"	38 38 39
Publish to SEWS Synchronously	40
Subscribe Asynchronously	42

# Chapter 6

Locating, Importing, and Using the Sample Projects	<b>43</b>
Sample Projects Overview	43
Locating and Importing the Sample Projects	44
Running the Sample Projects	44
Setting the Properties	45
Creating the Environment Profile	46
Deploying the Project	47

#### 4

Running the Sample	48
Bootstrap Invocation Parameters	48
Using the Sample Project in elnsight	49
eInsight Engine and eGate Components	49
The SWIFT_ADK_Sample_BPEL Sample Project	50
Sample Project Connectivity Map – cm_SWIFT_ADK	50
bp_SendToSWIFT BPEL Business Process	50
Data Used in the Sample Project	51
bp_ListenFromSWIFT BPEL Business Process	53
Using the Sample Project in eGate	56
Sample Project Connectivity Map – cm SWIFT ADK	56
jce_SendToSWIFT Collaboration Definition	57
SWIFT ADK eWay Error Messages	59

# Index

**62** 

# Chapter 1

# Introducing the SWIFT ADK eWay

This guide explains how to install, use, and operate the SeeBeyond® Integrated Composite Application Network Suite<sup>TM</sup> (ICAN) SWIFT ADK eWay Intelligent Adapter, referred to as the SWIFT ADK eWay throughout this guide.

This chapter provides a brief overview of operations, components, general features, and system requirements of the eWay.

#### **Chapter Topics**

- "SWIFT Overview" on page 6
- "SWIFT ADK eWay Overview" on page 7

## 1.1 SWIFT Overview

This section provides an overview of Society for Worldwide Interbank Financial Telecommunication (SWIFT) and the SWIFT ADK eWay.

#### Introduction to SWIFT

SWIFT is a bank-owned cooperative which 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) via its SWIFT Transport Network (STN). The types of events 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.

The SWIFT ADK eWay provides secure messaging services (both receiving and transmitting) between SWIFT financial institutions. The SWIFT ADK eWay is designed specifically to interface with the SWIFTAlliance, and enables the SeeBeyond eGate and eInsight system to exchange data with SWIFTAlliance by providing:

- Automated integration of securities events in the new securities standards (events MTxxx) which is based on the ISO15022 Data Dictionary.
- **Translation** of incoming events received from SWIFT into the format required by existing applications.
- **Security**, by enabling Secure Sockets Layer (SSL) between the SEWS executable and eGate.

The SWIFT ADK eWay uses the SWIFTAlliance Developer Toolkit (ADK), which is a library of APIs that can call services provided by SWIFTAlliance servers. For more information about ADK, see the *SWIFT ADK Reference Guide*.

# 1.2 SWIFT ADK eWay Overview

The SWIFT ADK eWay functions as a component within the SWIFTAlliance Access (SAA), operating in conjunction with the SWIFT ADK API to provide direct access to messages that pass between the SeeBeyond JMS IQ Manager and the SWIFT Network.

The SEWS Executable (SEWS.exe) component is linked into SAA through two routing points, one for incoming messages (SEWS\_from\_egate) and the other for outgoing messages (SEWS\_to\_egate).

The message remains in the routing point until the SEWS component forwards it to the JMS IQ Manager, or SAA delivers it to the network. The connection between eGate/eInsight and SWIFT can be monitored in the SWIFTAlliance log.

Both the eGate API Kit and the Swift ADK API transfer message data from the SAA to a Queue managed by the SeeBeyond JMS IQ Manager within the SeeBeyond Logical Host. JMS messages stemming from the JMS IQ Manager are either consumed or delivered transactionally. These transmissions occur between the JMS OTD, which resides in the Integration Server, and the Java Collaboration Definitions or BPEL business processes, which functions to transform inbound and outbound message data. The SWIFT ADK eWay supports both synchronous (request/reply) and asynchronous (send only) communication between these components.



#### Figure 1 SWIFT ADK eWay Message Flow Overview

## 1.2.1 Types of SWIFT ADK eWay Message Flows

Message that pass between SEWS and eGate/eInsight are classified within the following three message flows:

- Publish (outbound asynchronous)
- Request/Reply (outbound synchronous)
- Subscribe (inbound asynchronous)

### Publish (outbound asynchronous)

Outbound messages that are published asynchronously to SEWS are sent to the **SEWS\_from\_eGate** routing point without waiting to know the status or result of the request. We call the *send* method on the JMS OTD within a Java Collaboration or BPEL Business Process. This method does not return any value. The JMS message is sent to the JMS queue/topic (as defined by the Listening T/Q name). The same message is then retrieved by SEWS. Based on the message's "Header" property value, the SEWS component will either "PUT" or "LIST". The status/result of these actions can not be returned to the Java Collaboration because the JMS OTD *send* method is asynchronous. The result of "**PUT**" or "**LIST**" enters eGate through a JMS queue/topic (as defined by the Sending T/Q name). See Figure 2 for an illustration of the message transmission flow.

#### Figure 2 Publish (outbound asynchronous) Message Transmission Method



Publish (outbound asynchronous)

### Request / Reply (outbound synchronous)

When publishing outbound messages synchronously to SEWS, the message is sent to the **SEWS\_from\_eGate** routing point, then the Java Collaboration waits for the status or result of the request before sending another request. The synchronous nature is achieved by calling the JMS OTD method *requestReply*. This method is different from *send* in terms of its blocking/waiting. By calling this method in a Java Collaboration, a JMS message is sent to the JMS queue/topic (as defined by the Listening T/Q name). Also, the JMS OTD creates a hidden temporary JMS queue/topic for SEWS to reply to. Blocking/waiting happens when the JMS OTD becomes an immediate subscriber of the temporary queue/topic. In parallel, the SEWS component retrieves the message from the JMS queue/topic. SEW then processes the message by action defined in the JMS "Header" property. The result of the "PUT/LIST" action is sent as a JMS message by SEWS on the hidden temporary JMS queue/topic. The *requestReply* call unblocks and returns the resulting JMS message to a Java Collaboration. See Figure 3 for an illustration of the message transmission flow.

Figure 3 Request / Reply (outbound synchronous) Message Transmission Method



Request / Reply (outbound synchronous)

### Subscribe (inbound asynchronous)

Messages from SEWS always arrive at eGate/eInsight asynchronously on the JMS queue/topic (as defined by the Sending T/Q name). There can be three types of messages:

- The real SWIFT message from the SEWS\_to\_eGate routing point. This message leaves the routing point and enters eGate/eInsight as a JMS text message.
- The status message, if you publish to SEWS asynchronously.
- A heart beat message. This message implies that there is no available messages from the SEWS\_to\_eGate routing point.

See Figure 4 for an illustration of the message transmission flow.

#### Figure 4 Subscribe (inbound asynchronous) Message Transmission Method



## Subscribe (inbound asynchronous)

SWIFT message status also appears in outbound asynchronous message flows.

# **Chapter 2**

# Installing the SWIFT ADK eWay

This chapter describes the requirements and procedures for installing the SWIFT ADK eWay. Procedures for implementing sample projects, are described in **Chapter 6**.

#### **Chapter Topics**

- "Supported Operating Systems" on page 12
- "System Requirements" on page 12
- "External System Requirements" on page 13
- "Before Installing the eWay" on page 14
- "eWay Installation Procedures" on page 14
- "After Installation" on page 15

# 2.1 Supported Operating Systems

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

- Windows 2000
- IBM AIX 5.2
- Sun Solaris 9

# 2.2 System Requirements

To use the SWIFT ADK eWay, you need the following:

- An eGate Logical Host.
- A TCP/IP network connection to SWIFTAlliance.
- Sufficient free disk space to accommodate eWay files:
  - Approximately 200 KB on Windows systems
  - Approximately 820 KB on Solaris system
  - Approximately 500 KB on AIX systems

- Additional free disk space on the SWIFTAlliance host for the SEWS component (see additional information under External System Requirements on page 13):
  - Approximately 95 KB of disk space on Windows systems
  - Approximately 1.7 MB of disk space on Solaris systems
  - Approximately 780 KB of disk space on AIX systems
- *Note:* Additional disk space is required to process and queue the data that this eWay processes; the amount necessary varies, based on the type and size of the data being processed, and any external applications performing the processing.

#### **Logical Host Requirements**

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

#### **Environment Configuration**

No changes are required to the Logical Host's operating environment to support this eWay.

# 2.3 **External System Requirements**

- The SWIFT ADK eWay supports SWIFTAlliance version 5.5. The SWIFT ADK eWay also requires a SWIFT ADK eWay run-time license for SWIFTAlliance version 5.5.
- *Note:* The ADK (Alliance Developer Kit) API protocol is supported only by the SWIFTAlliance Access product family. It is **not** supported by SWIFTAlliance Entry. Customers using SWIFTAlliance Entry can send and receive SWIFT messages through the SeeBeyond Batch eWay, by appropriately configuring the AFT (Automated File Transfer) interface in SWIFTAlliance Entry.

#### **External Configuration Requirements**

- The SEWS component must be installed and configured (see Chapter 4)
- *Note:* The SEWS component must be installed on the same platform as the SWIFTAlliance server.
  - Before installing SEWS into SWIFTAlliance on UNIX, the root user must set up the correct environment.

Two routing points, **SEWS\_to\_egate** and **SEWS\_from\_egate** are installed when the **SEWS ADK** component is installed. The ways in which these routing points are used, and messages are routed to and from them, are independent of SEWS and depend on the application being used. For information on how to configure routing points, see the SWIFT *System Management Guide*.

# 2.4 **Before Installing the eWay**

Open and review the **Readme.txt** file for any additional information or requirements prior to installation. The readme.txt is located on the installation CD-ROM.

# 2.5 eWay Installation Procedures

During the ICAN Suite installation process, 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.

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

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

- 1 On the Enterprise Manager under the **ADMIN** tab, select the **SwiftADKeWay.sar** (to install the Swift ADK eWay) file to upload.
- 2 On the Enterprise Manager under the **Admin** tab, select the **FileeWay.sar** (to install the File eWay) file to upload.
- 3 On the Enterprise Manager under the Admin tab, select the **SwiftADKeWayDocs.sar** (to install the documentation and sample projects) file to upload.
- 4 On the Enterprise Manager under the **DOWNLOADS** tab, click the link for the SEWS component that matches the system running SAA. Files available for download include:
  - SEWS component for SAA5.5 Win32 downloads the SEWS\_Win32.zip file
  - SEWS component for SAA5.5 Aix52 downloads the SEWS\_AIX52.tar file
  - SEWS component for SAA5.5 Solaris 9 downloads the SEWS\_SOL9.tar file

You must save the file the following local directory folder:

<cd>:\SETUP\ADDONS\EWSWIFTADK\SEWS\WIN32\

These are SWIFT ADK medium files that are accessed directly from the SAA install GUI, (see **Installing and Setting up the SEWS eWay Component** on page 20).

5 On the Enterprise Manager under the **DOCUMENTATION** tab, click the sample project link. For the sample project, it is recommended that you extract the file to

another location prior to importing it using the Enterprise Explorer's Import Project tool.

For additional information on how to use eGate, please see the "eGate Integrator Tutorial".

# 2.6 After Installation

Once the 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.

# Chapter 3

# Configuring the SWIFT ADK eWay

This chapter explains how to set the properties for the SWIFT ADK eWay.

**Chapter Topics** 

- "SWIFT ADK eWay Configuration" on page 16
- "Configuring the SeeBeyond JMS IQ Manager" on page 17

# 3.1 SWIFT ADK eWay Configuration

The configuration settings within eGate for the SWIFT ADK eWay only apply to the SeeBeyond JMS IQ Manager. The SWIFT ADK eWay contains no eWay configuration properties since the SEWS component resides as a component within the SWIFTAlliance Access, which communicates directly with Topics of Queues managed by the SeeBeyond JMS IQ Manager.

#### To configure the SWIFT ADK eWay

- 1 From the Enterprise Designer, click the **Environment Explorer** tab located at the bottom left pane.
- 2 Create a new project **Environment**.
- 3 Create all the systems required to deploy your project, including:
  - External systems
  - Logical Host
  - SeeBeyond Integration Server
  - SeeBeyond JMS IQ Manager
- 4 Right-click **JMS IQ Manager** and select **Properties** from the list box.
- <sup>5</sup> The eWay **Properties dialog box** appears, see **Figure 5 on page 17**. You can use this dialog box to modify the **SeeBeyond JMS IQ Manager** properties necessary for communication with the **SEWS Executable** in the **SWIFTAlliance Access**.
- 6 Click **OK** then **Save All** to save your changes.

# 3.2 Configuring the SeeBeyond JMS IQ Manager

This chapter describes how to configure the SeeBeyond JMS IQ Manager properties from the Environment Explorer tab.

Properties you can configure on the dialog box include:

- Enable authentication and authorization
- Enable SSL
- Host Name
- Server Port
- Server SSL Port

Figure 5 SeeBeyond JMS IQ Manager Properties Dialog Box

P	roperties	
Configuration	¥ 12 IV  /=	
○ □ SeeBeyond JMS IQ Manage	Authenticate Mode	TrustAll
	Enable authentication and aut	None
	Enable SSL	False
	Host name	localhost
:	JVM shared library path	
	Server port	18004
	Server SSL port	18005
Description (JMS Connection Con JMS Connection Configuration Parameters		
; Comments (JMS Connection Con	Pronerties	
ОК	Cancel	

### Enable authentication and authorization

#### Description

Turns the authentication and authorization security service on or off. If **File** is selected, then the Sun Java System, AD and OpenLDAP access to the SeeBeyond JMS IQ Manager is only granted if a valid user id and password are supplied. If **File** is selected for File-based aa service, **SunJava** is selected for Sun Java System, **AD** is selected for Active Directory Service, **OpenLDAP** is selected for OpenLDAP Directory Service.

#### **Required Values**

Change the default value to **File**.

**Note:** Previous versions of eGate use the value **True** to require user id and password access through the SeeBeyond JMS IQ Manager. You can access the JMS configuration properties by double-clicking the JMS icon on the connectivity map. Security settings are found under: **Configuration** > **JMS Client** > **Basic** > **Security**. For additional information on these settings, refer to the eGate Integrator JMS Reference Guide.

### **Enable SSL**

#### Description

Turns on or off Secure Sockets Layer (SSL). TCP/IP connections from the client to the JMS IQ Manager are secured using SSL when this parameter is set to **True**, but you must use the Server SSL port to take advantage of it.

#### **Required Values**

The default value is **False**.

### Host Name

#### Description

Lists the host name of the machine running the SeeBeyond JMS IQ Manager.

#### **Required Values**

The default value is **localhost**.

### **Server Port**

#### Description

Specifies the TCP/IP port number that the SeeBeyond JMS IQ Manager listens on. Each SeeBeyond JMS IQ Manager must have a unique port number per Logical Host. Change this setting to an available port if you add additional SeeBeyond JMS IQ Managers.

#### **Required Values**

The value displayed in the Server Port field is different for each version of eGate installed.

### Server SSL Port

#### Description

Specifies the TCP/IP port number that the SeeBeyond JMS IQ Manager listens on for Secure Socket Listener (SSL) connections. This port number is enabled when you change the Enable SSL configuration property to "**True**".

#### **Required Values**

The value displayed in the Server SSL Port field is different for each version of eGate installed.

# Chapter 4

# Installing and Setting up the SEWS eWay Component

This chapter describes procedures for installing the SEWS component within the SWIFTAlliance Access, and for customizing the SEWS component to operate with your system.

#### **This Chapter Includes:**

- "Overview" on page 20
- "Installing the SEWS Component on Windows" on page 21
- "Installing the SEWS Component on UNIX" on page 23
- "Configuring SEWS" on page 26
- "Starting the SEWS Swift ADK eWay Component" on page 35
- "Setting Up a Test Environment (Optional)" on page 36

### 4.1 **Overview**

Prior to installation you must:

- Install the SEWS component into the SWIFTAlliance server
- Make sure all SWIFTAlliance servers are not running
- Click Cancel if the Add-ons installer starts up automatically

You must properly configure the SEWS component to connect SWIFTAlliance to the SeeBeyond JMS IQ Manager.

# 4.2 Installing the SEWS Component on Windows

#### To install the SEWS component on a Windows system

1 Unzip the SEWS component for the SAA5.5\_Win32 to a local folder:

<cd>:\temp

- 2 Log into SWIFTAlliance ADK as the user account under which it was installed.
- <sup>3</sup> Start the SWIFTAlliance ADK setup program by locating and running the **adk\_install.exe** on SAA 5.5. The SWIFTAlliance Set-up window appears.

SWIFTAlliance Set-up (init		
Component SEWS Software Install component Component Component Add patch	Loading 03:35 INFO Starting ADK Installation (v9) 03:35 INFO Disabling all_adm logins 03:35 INFO Database checked 03:35 INFO Lock set	
<ul> <li>◆ Remove patch</li> <li>Services</li> <li>◆ Register</li> <li>◆ Unregister</li> <li>Status</li> <li>◆ List components</li> </ul>	03:35INFOLataloging /export/home/alliance/SEMS.tar03:35INFORestoring tar_SEWS_2_0 (from SEWS.tar)03:35INFORestoring ALL files of tar_SEWS_2_003:35INFOSEWS.tar successfully validated03:35INFOChecking directory structure03:35INFOEnvironment ready for changes03:35INFOMoving SEWS software under ./rls/SEWS03:35INFOMoving ALL files of SEWS03:35INFOVestoring ALL files of SEWS03:35INFOSecuring component file03:35INFOSEWS component software installed03:35INFOADK log files updated in backup database03:35INFOADK log files updated in backup database03:35INFOADKI session completed	
Insuit deution		
Cipher *	****	
ОК	Cancel	

Figure 6 SWIFTAlliance Set-up (Install component)

- 4 In the Set-up dialog, install SEWS as follows:
  - A In the Component field, type SEWS.
  - **B** In the Software group, select the **Install component** option (or **Upgrade component**, if a previous version of SEWS is installed)
  - **C** In the **Input device** field, type in the path to the folder extracted in step 1:

<drive>\<temp directory>

- **D** In the Cipher field, enter "**seebeyond**"
- **E** Click **OK**. Progress messages display in the dialog box. SEWS is successfully installed when you see the following message:

ADKI session completed

#### Figure 7 SWIFTAlliance Set-up (Register)

SWIFTAlliance Set-up (init)		
Component SEWS		
Software	Loadino	
✓ Install component		
Remove component	04:55 INFU Starting HUK Installation (VS) 04:59 WARN Cannot find current version of SEWS database	
🗇 Add patch	04:59 INFO Disabling all_adm logins	
🗇 Remove patch	04:55 INFO Database checked	
Services	04:59 INFO Lock set 04:59 INFO Cataloging /export/home/alliance/SEMS tar	
🔶 Register	04:59 INFO Restoring tar_SEWS_2_0 (from SEWS_tar)	
	04:59 INFO Restoring ALL files of tar_SEWS_2_0 04:59 INFO Will register SEWS from version 0 to 2.0	
Status	04:59 INFO Environment ready for changes	
	04:59 INFO Starting registration 04:59 INFO Processing 'register comp'	
	04:59 INFO Can take some time to complete	
	INFU Starting registration program of SEWS Version 2.0. INFD adk install 'mode' is INSTALL.	
	INFO: Inserting into comp	
	INFU CUMP entity - 1 entry added. INFO: Inserting into appl	
	INFO APPL entity - 1 entries added.	
	INFU: Inserting into mpth INFO MPFN entity - 1 entry added.	
	INFO: Inserting into exsa	
	INFO EASH entity - I entry added. INFO: Inserting into evtd	
	INFO EVID entity - 3 entries added.	
	INFO; Inserting into rpoi INFO RPOI entity - 2 entries added.	
	INFO: Inserting into rule	
	INFO: Inserting into cnfg	
	INFO CNFG entity - 8 entries added.	
	INFO All entries added to entities.	
	04:59 INFO See /tmp/adki_err_24_Jun_528636 for more info on this products INFO Registration SUCCESSEN	ocess
	04:59 INFO Exiting	
	04:59 INFO ADK log files updated in backup database	
	04:59 INFO ADKI session completed	
Input device		
Cipher		
ov 1	Carrol	

- 5 If you are preforming a first-time installation, register SEWS as follows:
  - A In the **Set-up** dialog **Software** group, clear the **Install component** option by selecting **Register**.
  - **B** Clear the **Input device** and **Cipher** fields.
  - C Click **OK**. Progress messages are displayed in the message area of the dialog. SEWS is successfully registered when you see the following message:

ADKI session completed

D Click **Quit** to close the Set-up window.

# 4.3 Installing the SEWS Component on UNIX

To install the SEWS component on a UNIX system

- 1 Log in as root, running under /bin/ksh.
- 2 Save the SEWS component for **SAA5.5\_AIX52** or **SAA5.5\_Solaris\_9** to:

/export/home/alliance/

- <sup>3</sup> Set up the correct environment by sourcing the following script (including the period and space at the beginning):
  - . /usr/swa/alliance\_init -s
- 4 Set up the LIBPATH (for AIX 5.2) or the LD\_LIBRARY\_PATH (for Solaris 9) to:

LIBPATH=/usr/alliance/rls/common/lib/AIX

LD\_LIBRARY\_PATH=/usr/alliance/rls/common/lib/SunOS

- 5 Set the display variable.
- 6 Start the SWIFTAlliance ADK setup program by typing the following command:
  - \$ (ALLIANCE) /INA/bin/\$(ARCH)/adk\_install:

Note: On Solaris, \$(ARCH) is set to 'SunOS'; on AIX 5.2, \$(ARCH) is set to 'AIX'.

The **SWIFTAlliance Set-up** window appears.

∑S₩IFTAlliance Set-up (ir	it)	
Component SEWS		
Software Install component Upgrade component Remove component Add patch Remove patch Services Register Unregister Status List components	Loading 03:35 INFO Starting ADK Installation (v9) 03:35 INFO Disabling all_adm logins 03:35 INFO Database checked 03:35 INFO Lock set 03:35 INFO Cataloging /export/home/alliance/SEWS.tar 03:35 INFO Restoring tar_SEWS_2_0 (from SEWS.tar) 03:35 INFO Restoring ALL files of tar_SEWS_2_0 03:35 INFO Restoring ALL files of tar_SEWS_2_0 03:35 INFO Checking directory structure 03:35 INFO Checking directory structure 03:35 INFO Checking directory structure 03:35 INFO Checking directory structure 03:35 INFO Environment ready for changes 03:35 INFO Moving SEWS software under ./rls/SEWS 03:35 INFO Restoring ALL files of SEWS 03:35 INFO Restoring and the file 03:35 INFO Securing component files 03:35 INFO Securing component software installed 03:35 INFO ADK log files updated in backup database 03:35 INFO ADK log files updated in backup database 03:35 INFO ADKI session completed	
Input device	/alliance/SEWS_AIX52.tar	
Cipher	*****	
ОК	Cancel	

Figure 8 SWIFTAlliance Set-up (Install component)

- 7 In the Set-up dialog, install SEWS as follows:
  - A In the **Component** field, type SEWS.
  - **B** In the **Software** group, select the **Install component** option (or **Upgrade component**, if a previous version of SEWS is installed).
  - C In the **Input device** field, type in the fully-qualified path on the installation CD-ROM, such as:

<cd>:/export/home/alliance/SEWS\_AIX52.tar

- D In the Cipher field, enter "seebeyond"
- E Click **OK**. Progress messages display in the dialog box. SEWS is successfully installed when you see the following message:

ADKI session completed

SWIFTAlliance Set-up (ini	
Component SEWS	
Software	
Software Software Install component	Loading
	04:59 INFO Starting ADK Installation (v9)
V Remove component	04:59 INFO Disabling all adm loging
🗇 Add patch	04:59 INFO Checking database consistency
🗇 Remove patch	04:59 INFO Database checked
Services	04:59 INFO Lock set
🔶 Register	04:59 INFO Cataloging Zexpontzionezalilancezations.tan 04:59 INFO Restoring tan SEWS 2.0 (from SEWS.tan)
🗇 Unregister	04:59 INFO Restoring ALL files of tar_SEWS_2_0
Status	04:59 INFO Will register SEWS from version 0 to 2.0
🗇 List components	04:59 INFU Environment ready for changes 04:59 INFO Starting registration
• • • • • • • • • • • • • • • • • • • •	04:59 INFO Starting registration 04:59 INFO Processing (register comp)
	04:59 INFO Can take some time to complete
	INFO Starting registration program of SEWS Version 2.0.
	INFO adk_install moder is INSTALL.
	INFO: Inserting into comp INFO: COMP entity - 1 entry added.
	INFO: Inserting into appl
	INFO APPL entity - 1 entries added.
	INFO: Inserting into mpfn
	INFU MPFN entity - 1 entry added.
	INFO. EXSA entity - 1 entry added.
	INFO: Inserting into evtd
	INFO EVID entity - 3 entries added.
	INFU: Inserting into rpoi
	INFO REFORENCES added.
	INFO RULE entity - DEFAULT rules - 2 entries added.
	INFO: Inserting into cnfg
	INFO CNFG entity - 8 entries added.
	INFO SUCLESSFUL exit of the registration program.
	04:59 INFO See /tmp/adki err 24 Jun 528636 for more info on this process
	04:59 INFO Registration SUCCESSFUL
	04:59 INFO Exiting
	04:59 INFO ADK log files updated in backup database
	04:59 INFO LOCK UN-Set 04:59 INFO ADKI session completed
-	
Input device	alliance/SEWS_AIX52.tar
Cipher	
E L	
OK	Cancel Quit

#### Figure 9 SWIFTAlliance Set-up (Register)

- 8 Register SEWS as follows if you are performing a first-time installation:
  - A In the **Set-up** dialog **Software** group, clear the **Install component** option.
  - B In the Services group, select Register.
  - C Clear the **Input device** and **Cipher** fields.
  - D Click **OK**. Progress messages are displayed in the message area of the dialog. SEWS is successfully registered when you see the following message:

ADKI session completed

**E** Click **Quit** to close the Set-up window.

# 4.4 Configuring SEWS

To configure SEWS, first set your SWIFTAlliance servers to **housekeeping** mode, then set up the following item to match your system and your schema:

- Network Parameters
- *Note:* The values entered for the SEWS parameters must exactly match the corresponding eWay configuration parameters.

## 4.4.1 System Management Window

The System Management window lists the seven parameters that require configuration. Each parameter must contain the proper value or the SEWS executable will attempt to initiate a JMS connection to a nonexistent JMS server.

To open the System Management window:

1 Click the **System Management** icon on the **SWIFTAlliance Access Control** window. The System Management window appears.

System	Management								
File	<u>Configuration</u>	ueue <u>D</u> evice	View	<u>E</u> dit					Help
	Conf	iguration				Queue		Device	
Comp.	Class	Parameter				Name		Device name	
MXS	Batch Input	Automatic - I	Error Di	ir	4	BatchSwiftAcks	A	/dev/tty0	
MXS	Batch Input	Automatic – H	Polling	Timer		BatchSwiftNaks		/dev/tty1	
MXS	Batch Input	History Perio	bd			DeliveryNotifAcks			
MXS	Print	Skip Interve	ntions			DeliveryNotifNaks			
MXS	Print	ST200-like Fo	ormat			LocalSwiftAcks			
SEWS	Network paramet	IP Address o	r host r	name		LocalSwiftNaks			
SEWS	Network paramet	Listening TQ	Name			PccSwiftAcks			
SEWS	Network paramet	Message Serve	er Logir	n Name		PccSwiftNaks			
SEWS	Network paramet	Message Type				Received			
SEWS	Network paramet	Sending TQ Na	ame			SEWS_from_egate			
SEWS	Network paramet	TCP Port Num	ber			SEWS_to_egate			
SEWS	Network paramet	Use SSL Flag				Statement			
SIS	FIN Copy	Local 103 Add	dition			System			
SIS	Message	LT load bala	ncing			ToBeInvestigated			
SIS	Performance	Active Corres	spondent	:		UpdateKeys			
SIS	Message	RTV Routing				_AI_from_APPLI			
SIS	Network	SWIFTNet Bate	ching Ti	imeout	$\overline{\Delta}$	_MP_authorisation	V		

#### Figure 10 Systems Management Window

### 4.4.2 Network Parameters

The Systems Management window of SWIFTAlliance allows you to configure seven *non-secret* parameters (see **Figure 10 on page 26**) of the SEWS component, including:

- IP Address or host name
- Listening TQ Name
- Message Server Login Name
- Message Type
- Sending TQ name
- TCP Port Number
- Use SSL Flag

### IP Address or host name

1 Double-click the **IP Address or host name** line in the System Management Window, to open the configuration details for that parameter, (see Figure 11 below).

😹 System Management - Configurat	ion Details	
<u>F</u> ile <u>C</u> onfiguration		Help
Component	Value: portal-eway	
SEWS		
Class	Default value: 127/0/0/1	
Network parameters		
Parameter		
IP Address or host n		
Alliance	Description:	
Previous	The IP address or bost name of message server	A
Next	to bind to. Refer to the SEWS documentation for more information.	
Modify		Ā

Figure 11 Systems Management – IP Address

2 Enter the IP Address or host name where your message server runs.

Note: Review the SWIFT documentation for the required IP Address format.

### Listening TQ Name

3 Double-click the **Listening TQ Name** line to open the configuration details for that parameter, (see Figure 12 below).

🐱 System Management - Configuration Details		
<u>F</u> ile <u>C</u> onfiguration		Help
Component	Value: SEWS_from_egate_aix	
<b>ŠEMS</b>		
Class	Default value: SEWS_from_egate	
Network parameters		
Parameter		
Listening TQ Name		
Alliänce	Description:	
Previous	The incoming Topic or Queue name.	A
Next	Refer to the SEWS documentation for more information.	
Modify		V

#### Figure 12 Systems Management – Listening TQ Name

4 Enter the name of the JMS Topic or Queue in the **SeeBeyond Enterprise Designer – Connectivity Map Editor** that is designated for sending messages to SEWS.

The SEWS executable listens for available messages and inserts the SWIFT message to the **SEWS\_from\_egate** routing point. You can provide any name for the JMS queue or topic in eGate, but not for the routing point which is registered with SWIFT. We recommend providing similar naming conventions to distinguish between incoming and outgoing messages.

### Message Server Login

5 Double-click the **Message Server Login** line in the System Management Window, to open the configuration details for that parameter, (see Figure 13 below).

System Management - Configuration Details File Configuration Help Component Value: Administrator **SEMS** Default value: Administrator Class Network parameters Parameter Message Server Login Alliance Description: Previous The user login name to message server. Refer to the SEWS documentation for more Next information. Modify

Figure 13 Systems Management – Message Server Login

6 Enter the Message Server Login name to communicate with the JMS server.

This parameter only takes affect if you enable the JMS message server to authenticate the client. The example in Figure 13 uses the default value.

### Message Type

7 Double-click the **Message Type** line in the System Management Window, to open the configuration details for that parameter, (see Figure 13 below).

🛃 System Management - Configurat	ion Details	
<u>F</u> ile <u>C</u> onfiguration		Help
Component	Value: QUEUE	
SEWS		
Class	Default value: QUEUE	
Network parameters		
Parameter		
Message Type		
Alliance	Description:	
Previous	The Topic or Queue message tupe.	A
Next	Refer to the SEWS documentation for more information.	
Moxiify		X

Figure 14 Systems Management – Message Type

8 Enter **QUEUE** or **TOPIC** in upper case characters. Entering any other character causes the value to default to queue mode communication with the JMS message server.

## Sending TQ Name

9 Double-click the **Sending TQ Name** line in the System Management Window, to open the configuration details for that parameter, (see Figure 13 below).

System Management - Configuration Details File Configuration Help Component Value: SEWS\_to\_egate\_aix **ŠEMS** Default value: SEWS\_to\_egate Class Network parameters Parameter Sending TQ Name Alliance Description: Previous The outgoing Topic or Queue name. Refer to the SEWS documentation for more Next information. MoxILEy

Figure 15 Systems Management – Sending TQ Name

10 Enter the name of the JMS **Topic** or **Queue** in the **Connectivity Map Editor** that is designated for receiving messages from SEWS.

The SEWS Executable uses the JMS Topic or Queue name to subscribe to a message from the routing point, such as "**SEWS\_to\_egate**", and mashals the ADK message into a SWIFT message. The SEWS Executable then converts the SWIFT message to a text message and delivers it to the SeeBeyond IQ Manager. You also need to subscribe to the JMS Queue or Topic in the eGate Collaboration.

### **TCP Port Number**

11 Double-click the **TCP Port Number** line in the System Management Window, to open the configuration details for that parameter, (see Figure 13 below).

System Management - Configuration Details File Configuration Help Component Value: 18008 **SEWS** Default value: 14000 Class Network parameters Min allowed value: 1 Parameter Max allowed value: 65535 TCP Port Number Alliance Description: Previous The TCP Port number to listen on. Refer to the SEWS documentation for more Next information. Modilfy

Figure 16 Systems Management – TCP Port Number

12 Enter the port number the SeeBeyond IQ Manager uses to communicate to the JMS message client. You can get this value from the **JMS IQ Manager Properties** window in the Environment Explorer.

## **Use SSL Flag**

13 Double-click the **Use SSL Flag** line in the System Management Window, to open the configuration details for that parameter, (see Figure 13 below).

System Management - Configuration Details		
<u>F</u> ile <u>C</u> onfiguration	Help	
Component	Value: 0	
<b>ŠEMS</b>		
Class	Default value: 0	
Network parameters	Min allowed value: Ø	
Parameter	Max allowed value: 1	
Use SSL Flag		
Alliance	Description:	
Previous	FLAG to indicate connect the message sever port	
Next	using SSL. Refer to the SEWS documentation for more information.	
MoxII.fy		

Figure 17 Systems Management – Use SSL Flag

14 Enter an integer value of "1" if you want the SEWS executable to communicate with the JMS message server using Secured Sockets Layer (SSL) protocol. Otherwise, set the value to "0". If you set the value to "1", then the TCP Port Number in the previous section should be the SSL port used in the SeeBeyond JMS IQ Manager's Properties dialog box, located on the Environment Explorer.

# 4.4.3 Security Definition Window

The Security Definition window can only be run by security officers **LSO** and **RSO** (which stands for Left and Right Security Office, respectively). Any changes made by any security officer to any of the security definitions must be approved by both security officers before they can become effective.

🖪 Configuration - Security Definition 📃 🗖 🗙		
<u>File Edit Configuration View Options H</u> elp		
Class	Parameter	Approval Status
Signon	Timeout	Approved
Signoff	Timeout	Approved
Operator	Restrict Functions	Approved
Password	Master Period	Approved
Password	Max Bad Pwd	Approved
Password	Min Pwd Length	Approved
Password	Nbr Retained Pwd	Approved
Password	User Period	Approved
Password	Mode	Approved
Password Illegal Patterns		Approved
Password Reset Peer Officer Passwo		Approved
Reports Root Path for Report File		Approved
Signon Multiple		Approved
System	RPC Authentication	Approved
User Mode Housekeeping User Mode Approved		Approved
Security parameters	Secret	Approved
	÷	
For Help, press F1		

#### Figure 18 Security Definitions window

#### Secret

Enter the value that will be used to connect to eGate. You must first login as LSO to enter this value, then login as RSO to approve the value.

#### Figure 19 Security Definition window



# 4.5 Starting the SEWS Swift ADK eWay Component

#### To start the SEWS component

- 1 Set your SWIFTAlliance servers to **operational** mode.
- 2 Select **Start Component** from the **File** menu in the System Management window. The Start Component window appears.



Figure 20 System Management – Star Component

3 Select **SEWS Swift ADK eWay Server** from the component list and click **Start**.

# 4.6 Setting Up a Test Environment (Optional)

If you do not have an existing SWIFT environment suitable for testing, you can set up a simple loopback configuration for testing purposes, following installation of the SEWS component. When running this test environment, any eGate-outbound events sent to the SEWS component will be routed back as eGate-inbound events, using the **SEWS\_from\_egate** and **SEWS\_to\_egate** routing points.

To create this test environment, you must set up the following items to match your system and your schema.

- 1 Queues for the routing points.
- 2 The routing points themselves.
- 3 Logical terminals.
- 4 TCP/IP parameters.
- 5 Security options.

Remember to set your SWIFTAlliance servers to **housekeeping** mode. All procedures except those related to security options require only Operator privileges.

*Note:* When you create a new security, both the LSO and RSO must log in and *approve* it before you can *activate* it.

### 4.6.1 **Queues**

First, you need to set up the queue for the routing points:

- 1 Launch the Systems Management window, invoke the **View** drop-down menu, and select **Queue**.
- 2 Open the routing point **SEWS\_from\_egate**, and select the Routing tab.
- 3 Under *Valid routing targets,* move **SEWS\_to\_egate** from **Available** to **Selected**.
- 4 Modify and Save this configuration.

### 4.6.2 Routing Points

Second, you need to configure the routing points to loop the messages from *inbound* back to *outbound*:

- 1 Launch the Routing window.
- 2 View the schemas to make sure your test schema is separate from any working schema. If not, create a dummy schema and move it to the **Selected** box under the **Used in Routing Schema(s)** field.

<sup>3</sup> With the test schema active, open the routing point **SEWS\_from\_egate** and create a new routing rule, setting the following parameters to the indicated values.

Parameter	Value
Condition On	Function Result and Message
Function Result	Accept
Message	Creating_mpfn = 'SEWS_mpf'
Action On	source
Action	Route To (on the source sub-panel)
Route To	SEWS_to_egate
Append Intervention	No Intervention
Unit	Keep Current

- 4 Click **Validate**, then **Save** the configuration.
- 5 View the **Routing Rule** window and click **Add** or **Modify** to save the routing rule settings.

# 4.7 Authentication

The SWIFT ADK eWay only supports password based authentication. The SEWS component authenticates itself to the eGate IQ Manager by providing **Message Server Login** (see **Message Server Login** on page 29) and **Secret** (see **Secret** on page 34). For additional security, we recommend enabling SSL. The SEWS component always trusts the eGate IQ Manager. Server authorization (IQ Manager authenticating itself to SEWS) is currently not supported.

# **Chapter 5**

# Using JMS OTD with the SEWS Component

This chapter describes the properties used to establish message connections between the JMS OTD and the SEWS Component.

This Chapter Includes:

- "Publish to SEWS Asynchronously" on page 38
- "Publish to SEWS Synchronously" on page 40
- "Subscribe Asynchronously" on page 42

# 5.1 **Publish to SEWS Asynchronously**

This section describes the steps required to set up JMS properties that are required to publish to SEWS asynchronously (publish – outbound asynchronous). This component supports two types of messages:

- PUT
- LIST

For additional information on how these message types are implemented in a typical outbound asynchronous message flow, see **Publish (outbound asynchronous) Message Transmission Method** on page 9.

### 5.1.1 Setting the JMS Header property as "PUT"

To insert a SWIFT ADK eWay message using a Java Collaboration or BPEL business process, you must:

1 Create a text message that invokes the JMS OTD's *createTestMessage* method. In a Java Collaboration, this method appears as:

```
com.stc.connectors.jms.Message requestTextMsg =
JMS_1.createTextMessage();
```

- 2 Call the *storeUserProperty* method on the requestTextMsg object to set up the following JMS properties:
  - **storeUserProperty("Header", "PUT")**, the "PUT" means the SEWS will put the message into routing point SEWS\_from\_egate.
  - storeUserProperty("type", "Swift"), means the message type is Swift.
  - **storeUserProperty("validation", "none")**, the level of message validation you want SAA to do it before insert to routing point. SAA supports these possible validation levels:
    - none
    - minimum
    - intermediate
    - maximum

If you misspell the validation property or do not set validation property, the SEWS takes the default validation level: "maximum".

- storeUserProperty("recourse\_action","complete"). The possible choices are:
  - **complete**: means the message will be complete if validation ok
  - modify: this means the message will be modified due to validation failure.
- **storeUserProperty("duplicate","true")**, you only set this user property when you know the message you want to insert into RP: SEWS\_from\_egate is a duplicated message.
- Set the text pay load on this JMS OTD object.

## 5.1.2 Setting the JMS Header property as "LIST"

Another "Header" action is "**LIST**", which allows you to get a list of messages on the routing point. The following JMS properties apply:

- **storeUserProperty("Header", "LIST")**, the "LIST" means the SEWS will list the messages on the RP.
- **storeUserProperty("state", "reserved")**, it will only list "reserved" messages, choices are:
  - all
  - reserved
  - unreserved

If you do not set "state" property, the default state is "unreserved".

- storeUserProperty("rp\_name", "SEWS\_to\_egate"), which RP you want to list, choices are:
  - SEWS\_to\_egate
  - SEWS\_from\_egate.

Publishing to SEWS asynchronously requires calling the *send* method on this OTD. If you use eGate 5.0.1 – 5.0.3, the JMS OTD does not have a *requestReply* method, so you can only call *send*. The result of "**PUT**" or "**LIST**" will come in eGate through the JMS queue/topic (as defined in the Sending T/Q name).

# 5.2 Publish to SEWS Synchronously

This section describes the steps required to set up JMS properties that are required to publish to SEWS synchronously (Request / Reply – outbound synchronous).

For additional information on how these message types are implemented in a typical outbound synchronous message flow, see **Request / Reply (outbound synchronous) Message Transmission Method** on page 10.

This feature is only supported in eGate 5.0.4 with the addition of a new method in the JMS OTD.

Everything is the same as in "Publish to SEWS asynchronously", except for the last step. Instead of calling the *send* method, you call *requestReply(requestTextMsg)*;.

There will be a return result from calling *requestReply*, the return result is a com.stc.connectors.jms.Message type object. You should call *retrieveUserPropertyList* on the reply object to get all the user properties. Also you should call *retrieveStringFromMessage* to retrieve the reply message body.

If your request is "**Header=PUT**", then the reply message is a status report of "**PUT**". The reply message has the following JMS properties:

- **s\_umid**, the value of this JMS property is the s\_umid of new swift message we add to RP: SEWS\_from\_egate.
- **JMS\_ProducerID**, the value is always ICANSEWSLSTN.
- isOK, the value if "TRUE" if you insert successfully, otherwise "FALSE".
- times, the value a series of timestamp the "PUT" request take to put a message.

If "**isOK=FLASE**", e.g. the message has the wrong format. The JMS property will list the cause of the failure, it will has the following properties:

- code, value may be "SEWS\_FAILED\_TO\_ADD". Other possible codes are:
  - SEWS\_UNKNOWN\_COMMAND
  - SEWS\_UNKNOWN\_STATE
  - SEWS\_FAILED\_TO\_ROUTE
  - SEWS\_FAILED\_TO\_RESERVE
  - SEWS\_UNKNOWN\_MSG\_TYPE
  - SEWS\_NO\_MSG\_TYPE\_SUPPLIED
  - SEWS\_INVALID\_RP.
- JMS\_ProducerID, same as previous.
- isOK, "FALSE"
- **reason**, value is "ADK\_FAILURE

If your request is "**Header=LIST**", then the reply message is a list of messages' **s\_umids** The reply message has JMS properties:

- isOK, "TRUE" or "FALSE"
- JMS\_ProducerID, the value is always ICANSEWSLSTN
- **code**, value may be "SEWS\_FAILED\_TO\_LIST". Other possible codes are:
  - SEWS\_UNKNOWN\_COMMAND
  - SEWS\_UNKNOWN\_STATE
  - SEWS\_FAILED\_TO\_ROUTE
  - SEWS\_FAILED\_TO\_RESERVE
  - SEWS\_UNKNOWN\_MSG\_TYPE
  - SEWS\_NO\_MSG\_TYPE\_SUPPLIED
  - SEWS\_INVALID\_RP.

You get code only isOK=FALSE.

The reply message body will have a list of **s\_umids** separated by space. In case there is no message, then the reply message body is a empty string, i.e. "".

# 5.3 Subscribe Asynchronously

This section describes the steps required to set up JMS properties that are required to subscribe to messages asynchronously (Subscribe – inbound asynchronous).

For additional information on how these message types are implemented in a typical inbound asynchronous message flow, see **Subscribe (inbound asynchronous) Message Transmission Method** on page 11.

These messages are delivered to eGate asynchronously from SEWS. There can be three types of messages:

- 1 The real swift message from RP: SEWS\_to\_egate, those messages are real SWIFT message and coming into egate as a JMS text message. The SWIFT message is contained in the JMS message body. It has following JMS message properties
  - mesg\_s\_umid: s\_umid of the message
  - **inst\_type:** always "ADK\_INST\_TYPE\_ORIGINAL", we only support routing original message
  - **inst\_num**, message instance number, normally 0 because we only support routing original message.
  - times, a series of timing data
  - mesg\_sub\_format, either "ADK\_INPUT" or "ADK\_OUTPUT"
  - JMS\_ProducerID
  - **inst\_s\_umid**, the instance s\_umid, normally same as mesg\_s\_umid.
  - isOK, always "TRUE"
  - type, can be "Swift" or "Telex"
  - text\_s\_umid, the message text body s\_umid, normally same as mesg\_s\_umid
  - **s\_umid**, the message body s\_umid, normally same as mesg\_s\_umid

The text payload of the JMS message will have the real Swift message without any string padding.

- <sup>2</sup> The status message, if you use "Publish to SEWS asynchronously", the message is a JMS message, has same format as we discussed in "Publish to SEWS synchronously", except the JMS\_ProducerID has new value: ICANFROMSEWS.
- 3 A heart beat message saying there is no message to get from RP: SEWS\_to\_egate. It has JMS properties:
  - code, "SEWS\_NO\_MESSAGE"
  - JMS\_ProducerID, "ICANFROMSEWS"
  - isOK, "FALSE". It is always FALSE because no message on RP: SEWS\_to\_egate, you should also look at "code" property for "SEWS\_NO\_MESSAGE" to filter out such benign heart beat message.
  - rp\_name, always "SEWS\_to\_egate"

# **Chapter 6**

# Locating, Importing, and Using the Sample Projects

This chapter describes how to use the sample projects included in the installation CD-ROM package.

**Chapter Topics Include:** 

- "Sample Projects Overview" on page 43
- "Locating and Importing the Sample Projects" on page 44
- "Running the Sample Projects" on page 44
- "Using the Sample Project in eInsight" on page 49
- "Using the Sample Project in eGate" on page 56

# 6.1 Sample Projects Overview

Sample projects are designed to provide an overview of the basic functionality of the SWIFT ADK eWay by demonstrating how to pass information between the SWIFTAlliance Access and eGate.

#### Sample Projects Include:

**SWIFT\_ADK\_Sample\_JCE** – uses Collaboration Definitions (Java) to either listen for, or send messages to SWIFT.

The following components are found in the sample project:

- Connectivity Map (cm\_SWIFT\_ADK)
- File External Applications (FileIN and FileOUT)
- Java based Collaboration Definitions (jce\_ListenFromSwift and jce\_SendToSwift)
- Queues (SEWS\_from\_egate and SEWS\_to\_egate)

**SWIFT\_ADK\_Sample\_BPEL** – uses BPEL based business processes that are designed to either listen for, or send messages to SWIFT.

The following components are found in the sample project:

- Connectivity Map (cm\_SWIFT\_ADK)
- File External Applications (FileIN and FileOUT)
- BPEL based business processes (bp\_ListenFromSwift and bp\_SendToSwift)
- DTD based OTD (UserPropertyMsg\_UserPropertyMsg)
- Queues (SEWS\_from\_egate and SEWS\_to\_egate)

# 6.2 Locating and Importing the Sample Projects

The eWay sample projects are included in the SWIFTADKeWayDocs.sar. This file is uploaded separately from the SWIFTADKeWay.sar file during installation. For more information, refer to "Installing the SWIFT ADK eWay" on page 12.

Once you have uploaded the **SWIFTADKeWayDocs.sar** to the Repository, you can begin downloading the sample Projects from the **Documentation** tab on Enterprise Manager, to a folder of your choosing.

Before using the sample project, you must first import it into the SeeBeyond Enterprise Designer using the Enterprise Designer Project Import utility.

#### To Import the Sample Project:

- 1 From the Enterprise Designer's Project Explorer pane, right-click the Repository and select **Import**.
- 2 In the **Import Manager** window, browse to the directory that contains the sample Project zip file.
- 3 Select the sample file and then click **Open**.
- 4 Click the **Import** button. If the import was successful, then click the **OK** button on the **Import Status** window.

# 6.3 **Running the Sample Projects**

Steps required to run the sample projects include:

- Configure the Properties
- Creating the Environment Profile
- Deploying the Project
- Running the Sample

# 6.3.1 Setting the Properties

Sample projects use both an inbound and an outbound File eWay to deliver or receive sample data. The following information describes how to configure the File eWays from the Connectivity Map.

#### To Configure the File eWays:

1 On the Connectivity Map, double-click the **Inbound File eWay** (denoted as the eWay that delivers data to the Collaboration Definition or BPEL Business Process).





2 The **Properties** window for the Inbound File eWay opens. Modify any parameter settings necessary for your system, including the **Directory** and **Input file name** so they match the location and name of the sample data file, see **Data Used in the Sample Project** on page 51.

Properties 🛛 🔀		
Configuration		
	Directory	C:/temp
	Input file name	input*.txt
	Input type	Bytes
·	Maximum bytes per record	4096
Description (parameter-settings)	Multiple records per file	False
	Polling interval	5000
	Remove EOL	False
Comments (parameter-settings)	Properties	
ОК	Cance	

Figure 22 Properties Dialog Box – Inbound File eWay

- 3 Click **OK** to close the **Properties** window.
- 4 On the Connectivity Map, double-click the **Outbound File eWay** (denoted as the eWay that receives data to the Collaboration Definition or BPEL Business Process).



Figure 23 Connectivity Map – Outbound File eWay

# SEWS\_to\_egate Service2 FileOUT

5 The Properties window for the Outbound File eWay opens. Modify any parameter settings necessary for your system, including the Output file name and the target Directory where you want the resulting data to appear.

Figure 24 Properties Dialog Box – Outbound File eWay

	Properties	8
Configuration		
Parameter Settings	Add EOL	True
	Directory	C:/temp
	Multiple records per file	True
	Output file name	output%d.dat
Description (parameter-settings) Parameter Settings subsection Comments (parameter-settings)		
	Properties	
ОК	Car	ncel

6 Click **OK** to close the Properties window.

# 6.3.2. Creating the Environment Profile

An eGate Environment represents the physical system required to implement a Project. A typical Environment contains several components, including Logical Hosts, Integration Servers, JMS IQ Managers, and External Systems. Environments are created using the Enterprise Designer's Environment Explorer.

#### To Create a New Environment for the Sample Projects:

1 On the Environment Explorer, right-click the **Repository** icon and select **New Environment**. This creates a new Environment that you can rename to match the sample project you want to deploy.

- 2 Right-click the new Environment and add the following components:
  - New Logical Host
  - New File External System (as an outbound eWay)
  - New File External System (as an inbound eWay)
- 3 Right-click the new Logical Host and add the following components:
  - New SeeBeyond Integration Server
  - New SeeBeyond JMS IQ Manager

**Figure 25 on page 47** shows an example of these Environment containers in the Environment Editor.



Figure 25 Environment Editor – Environment Containers

# 6.3.3 Deploying the Project

Deployment Profile maps the ICAN components with the Environment hosts (hosts) that support them. A project may have one or more Deployment Profiles, but each project's active Deployment Profile must reside within a separate Environment.

#### To create a Deployment Profile:

- 1 Right-click the sample project and select **New > Deployment Profile**.
- 2 In the **Create Deployment Profile** window, enter a name for the component and select the applicable **Environment** from the drop-down list. Click **OK** to close the window.

- <sup>3</sup> From the Deployment Profile pane, drag-and-drop all the ICAN icons from the left pane, to the applicable Environment containers on the right pane (See Figure 26 on page 48).
- 4 Click Activate to activate your Deployment Profile.



Figure 26 Dragging ICAN Icons to the Environment Containers

# 6.3.4. Running the Sample

Running a sample project requires starting the "Bootstrap" process. The Bootstrap is a process that launches the Integration Servers and Message Servers for one Logical Host. The Bootstrap process is started and stopped with the following scripts:

/ican50/logicalhost/bootstrap/bin/bootstrap.bat

/ican50/logicalhost/bootstrap/bin/shutdown.bat

When the Bootstrap is started it locates the Respository via the command line parameters or with the user-defined configuration file, **logical-host.properties**, located in:

/ican50/logicalhost/bootstrap/config

### **Bootstrap Invocation Parameters**

The following table lists the parameters required to invoke the bootstrap. You can enter these parameters in the **logical-host.properties** file, or create a command line that runs these parameters in a separate **.bat** file.

Parameter	Description	Example
-r	Repository URL	http://localhost:12000/Repository
-е	Environment	Environment1
-	Logical Host	LogicalHost1
-i	User Name	Administrator
-р	Password	STC

#### Table 1 Bootstrap Invocation Parameters

After completing the process, the Output file in the target directory—that was previously configured in the Outbound File eWay—contains all records retrieved from the database in text format.

# 6.4 Using the Sample Project in elnsight

This section describes in greater detail how the **SWIFT\_ADK\_Sample\_BPEL** sample project works with the ICAN Suite's eInsight Business Process Manager and the Web Services interface. This section *does not* explain how to create a project using an eInsight business process. For these instructions, refer to the *"eInsight Business Process Manager User's Guide"*.

Before running this sample project, you must:

- Import the sample Project, (see "Locating and Importing the Sample Projects" on page 44)
- Configure the properties, (see "Setting the Properties" on page 45)
- Create an Environment for the sample project, (see "Creating the Environment Profile" on page 46)
- Create a Deployment Profile, (see "Deploying the Project" on page 47)

# 6.4.1 eInsight Engine and eGate Components

You can deploy an eGate component as an Activity in an eInsight Business Process. Once you have associated the desired component with an Activity, the eInsight engine can invoke it using a Web Services interface. Examples of eGate components that can interface with eInsight in this way are:

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

Using the eGate Enterprise Designer and eInsight, you can add an Activity to a Business Process, then associate that Activity—such as an eWay—with an eGate component. When eInsight runs the Business Process, it automatically invokes that component via its Web Services interface.

# 6.4.2 The SWIFT\_ADK\_Sample\_BPEL Sample Project

The **SWIFT\_ADK\_Sample\_BPEL** sample Project uses two BPEL business processes to send data to, and receive data from the SWIFTAlliance Access. The flow of data between eGate and SWIFT are represented on the sample project **Connectivity Map**.

### Sample Project Connectivity Map – cm\_SWIFT\_ADK

A Connectivity Map provides a canvas for assembling and configuring a Project's components. The icons on the Connectivity Map Editor tool bar represent the components used to populate the Connectivity Map canvas. When linked together, the components define the Connectivity Map for your Project.

The **cm\_SWIFT\_ADK** Connectivity Map contains two components. The first component displays the flow of data from eGate to SWIFT, and uses a BPEL business process called **bp\_SendToSWIFT**, located in **Service1**. The second component displays the flow of data from SWIFT to eGate and uses a BPEL business process called **bp\_ListenFromSWIFT**, located in **Service2**. Both of these Connectivity Map components are seen below in Figure 27.

Figure 27 Components on the cm\_SWIFT\_ADK Connectivity Map



## bp\_SendToSWIFT BPEL Business Process

The **bp\_SendToSWIFT** BPEL business process is designed to pass data from a sample text file, using the **FileClient.receive** web service, to the Message Properties OTD, using the **SEWS\_from.JMS.send**.

#### Figure 28 bp\_SendToSWIFT BPEL Business Process



In addition to passing through the entire contents of the text file, The **SWIFT ADK API** also requires four string literals that the business process passes in as message properties, as seen in Figure 29.

Business Rule Designer	
Conversion 🐐 Datetime 🍬 Operator 🍬 Boolean 🍬 String 🎽 Nodes 🎽 Number 🍬	
Output All Input All	].
Business Process Attributes FileClient.receive.Output A string literal A	
bp_SendToSWIFT	

Figure 29 Business Rules in bp\_SendToSWIFT

### Data Used in the Sample Project

The data used for the inbound samples (data sent to SWIFT) is received from a text file included in the sample project called **Sample\_SWIFT\_Msg.txt**, see Figure 30 below.

```
Figure 30 Sample Data sent to SWIFTAlliance Access
```

```
{1:F01PTSAUSZZAXXX1515293255} {2:I5020500020701PTSAUSZZ80A082645987402
510130500S} {3: {108:32516200} } {4:
:16R:GENL
:20C::SEME//UESS420701B02216
:23G:NEWM
```

:98C::PREP//20020701152928 :22F::TRTR//TRAD :16R:LINK :20C::MAST//FUNDS502 :16S:LINK :16R:LINK :20C::PREV//FUNDS502.1 :16S:LINK :16S:GENL :16R:ORDRDET :94B::TRAD//EXCH/XPARFRPPBIC :22H::BUSE//SELL :22F::TRCN//SOLI :22F::CAOP//CASH :22F::TOOR//MAKT :22F::TILI//GTCA :22H::PAYM//APMT :98A::EXPI//20040731 :11A::FXIS//EUR :16R:TRADPRTY :95Q::SELL//DBZEFRPPXXX :16S:TRADPRTY :36B::ORDR//UNIT/100, :19A::ORDR//EUR1000, :35B:ISIN FR0000127771 ACT. VIVENDI UNIVERSAL /TS/EX/FRA :16R:FIA :22F::FORM//BEAR :16S:FIA :16S:ORDRDET :16R:SETDET :22F::SETR//TRAD :16R:AMT :19A::EXEC//EUR123, :16S:AMT :16R:AMT :19A::CHAR//EUR123, :16S:AMT :16R:AMT :19A::LOCO//EUR123, :16S:AMT :16S:SETDET - }

### bp\_ListenFromSWIFT BPEL Business Process

The **bp\_ListenFromSWIFT** BPEL business process is designed to listen for messages from SWIFT. Messages received by the business process first go through a Decision Activity to determine if the data is either valid or invalid. Valid data gets transformed and marshalled using a DTD based OTD before being written out to a text file. Invalid data passes to an empty Activity, see Figure 31 below.





The following BPEL based Activities are used in the business process:

- SEWS\_to\_egate.JMS.receive
- Branching Decision
- Business Rule
- UserPropertyMsg\_UserPropertyMsg.marshal

#### SEWS\_to\_egate.JMS.receive

The **SEWS\_to\_egate.JMS.receive** contains a JMS receive OTD, that uses JMS much like a web service to pull data into a queue. Figure 32 below lists the methods used.

Figure 32 SEWS\_to\_egate.JMS.receive – Receive OTD



#### **Branching Decision**

The Branching Decision (Case1) is used to create a condition that looks for text in a receiving file that starts with "{1:". Text that matches this criteria passes onto a **Business Rule** Activity. Text that fails to match this criteria pass into an **Empty** Activity, see Figure 33.

Name: Decision	Decision Gate Properties
Order of Execution  Order Link Condition  Case 1  Starts-with(getContainerData(JMS receive.Output', Message', 'TrextM  If no link conditions are true, use this default link:  Throw Exception  Link Condition  Link Case 1  Conversion * Datetime * Operator * Boolean * String * Nodes * Number *  Business Process Attributes  Message  Message Type  Hesute file Clientwite Input  File Clientw	Name: Decision
Order Link     Case 1   If no link conditions are true, use this default link:   Case 2   If expression evaluation fails: Throw Exception   Link Condition   Link Condition   Link:   Case 1   Conversion * Datetime * Operator * Boolean * String * Nodes * Number *   Business Process Attributes   Files/Properties   Wassage   TextMessage   Yit:   TextMessage   Yit:   TextMessage   Yit:   TextMessage TextMess	Order of Execution
1       Case 1         If no link conditions are true, use this default link:       Case 2         If expression evaluation fails:       Throw Exception         Link Condition       Intervention         Link:       Case 1         Conversion * Datetime * Operator * Boolean * String * Nodes * Number *         Business Process Attributes         Message         JMS.teceive.Output         Message Type         DytestMessage         StreamMessage	Order Link Condition
If no link conditions are true, use this default link: Case 2 If expression evaluation fails: Throw Exception Link: Case 1 Conversion * Datetime * Operator * Boolean * String * Nodes * Number * Business Process Attributes Business Process Attributes Business Process Attributes Business Process Attributes Image	1 Case 1 starts-with(getContainerData('JMS.receive.Output', 'Message', '/TextM
If no link conditions are true, use this default link: Throw Exception T Link Condition Link: Case 1 Conversion * Datetime * Operator * Boolean * String * Nodes * Number * Business Process Attributes Business Process Attributes Message Message Message Type BytestMessage Stream Message Stream Message Stream Message MapMessage TextMessage Stream Mapsage MapMess	
If expression evaluation fails:	If no link conditions are true, use this default link: Case 2
Link: Case 1 Conversion * Datetime * Operator * Boolean * String * Nodes * Number * Business Process Attributes Business Process Attributes Business Process Attributes Message Business Property Message Business Property String Iteral TextMessage TextMessage StreamMessage FileClient.write.Input	If expression evaluation fails: Throw Exception
Link: Case 1 Conversion * Datetime * Operator * Boolean * String * Nodes * Number * Business Process Attributes JMS.receive. Output Message Message JMSMessage JMSMessage JMSMessage String literal TextMessage StreamMessage FileClientwrite. Input	Link Condition
Conversion & Datetime & Operator & Boolean & String & Nodes & Number & Business Process Attributes Message Message JMS.receive.Output Message JMSMessageType A string literal TextMessage TextMessage FileClient.write.Inout	Link: Case 1
Business Process Attributes JMS.receive. Dutput Message Message Message Type JMSMessage Type BytesMessage TextMessage FileClient.write.Inout	Conversion & Datetime & Operator & Boolean & String & Nodes & Number &
JMS.receive.Output Message WiserProperty String1 String2 return boolean TextMessage StreamMessage FileClient.write.Input	Business Process Attributes
Image: String I     Image: String	∲− I JMS.receive.Output
A string literal String2 YI: Yetum boolean Yetum boolean	UserProperty string1
BytesMessage     Y1:     retum boolean     StreamMessage     MapMessage     FileClient.write.Inout	Image: Construction of the string interal     Image: Construction of the string interal       Image: Construction of the string interal     Image: Construction of the string interal
StreamMessage StreamMessage FileClientwrite.Inout	BytesMessage     (1:
Contraction of the second seco	- StreamMessage
	©− 🚟 MapMessage ⊙− ■ 🖫 FileClient write. Input
OK Apply Cancel	OK Apply Cancel

#### Figure 33 Branching Decision – Case1

#### **Business Rule**

The Business Rule is an Activity that passes the text message and the **Name** and **Value** user properties into the **UserPropertyMsg\_UserPropertyMsg**, see Figure 34.

Business Rule Designer	
🖇 Conversion 🐐 Datetime 🐐 Operator 🐐 Boolean 🐐 String 🐐	Nodes 🐐 Number 🐐
Output All	Input All
<ul> <li>Business Process Attributes</li> <li>JMS.receive.Output</li> <li>Message</li> <li>Warsage</li> <li>Value</li> <li>MessageProperties</li> <li>JMSMessage</li> <li>StreamMessage</li> <li>StreamMessage</li> <li>MapMessage</li> </ul>	Business Process Attribute UserPropertyMsg_UserPropertyMsg.marshal.Input UserPropertyMsg UserProperty Name ? Value ? TextMessage ?
FileClient.write.Input	

#### Figure 34 Business Rule Activity



The **UserPropertyMsg\_UserPropertyMsg.marshal** is a DTD OTD based Activity that marshals the contents of the message before passing it onto the **FileClient.write** Activity. Figure 35 below lists the methods used by this OTD.





#### FileClient.write

The **FileClient.write** is an Activity based on the File eWay's web service OTD, see Figure 36.

#### Figure 36 File eWay Write Methods



# 6.5 Using the Sample Project in eGate

This section describes in greater detail how the **SWIFT\_ADK\_Sample\_JCE** sample project works with Java based Collaboration Definitions created with the Collaboration Editor. This section *does not* explain how to create Collaborations in eGate. For these instructions, refer to the *"eGate Integrator User's Guide"*.

Before running a sample Project, you must:

- Import the sample Project, (see "Locating and Importing the Sample Projects" on page 44)
- Configure the eWay properties, (see "Setting the Properties" on page 45)
- Create an Environment for the sample project, (see "Creating the Environment Profile" on page 46)
- Create a Deployment Profile, (see "Deploying the Project" on page 47)

### Sample Project Connectivity Map – cm\_SWIFT\_ADK

A Connectivity Map provides a canvas for assembling and configuring a Project's components. The icons on the Connectivity Map Editor tool bar represent the components used to populate the Connectivity Map canvas. When linked together, the components define the Connectivity Map for your Project.

The **cm\_SWIFT\_ADK** Connectivity Map contains two components. The first component displays the flow of data from eGate to SWIFT, and uses a Java based Collaboration Definition called **jce\_SendToSWIFT**, located in **Service1**. The second component displays the flow of data from SWIFT to eGate and uses a Java based Collaboration Definition called **jce\_ListenFromSWIFT**, located in **Service2**. Both of these Connectivity Map components are seen below in Figure 27.

#### Figure 37 Components on the cm\_SWIFT\_ADK Connectivity Map



### jce\_SendToSWIFT Collaboration Definition

The jce\_SendToSWIFT Collaboration Definition is designed to pass data from a sample text file, using the FileClient.receive web service, to the Message Properties OTD, using the SEWS\_from.JMS.send. The following four user properties are also included with the text file using the storeUserProperty() method:

- storeUserProperty[1]: "Header", "PUT"
- storeUserProperty[1]: "type", "Swift"
- storeUserProperty[1]: "validation", "maximum"
- storeUserProperty[1]: "recourse\_action", "complete"

Figure 38 below lists the business rules used to create the **jce\_SendToSWIFT** Collaboration Definition.

Figure 38 jce\_SendToSWIFT Collaboration Definition Business Rules



## jce\_ListenFromSWIFT Collaboration Definition

The **bp\_ListenFromSWIFT** Collaboration Definition is designed to listen for messages from SWIFT. When a messages arrive into the JMS Queue, the Collaboration Definition first checks to see if the text of the message starts with "{1:". If the message is valid, then **Name** and **Value** parameters are passed into variables that are concatenated together and written out as a text message.

Figure 39 below lists the business rules used to create the jce\_ListenFromSWIFT Collaboration Definition.

Figure 39 jce\_ListenFromSWIFT Collaboration Definition Business Rules



# 6.6 SWIFT ADK eWay Error Messages

The following error messages can be returned:

Content	Meaning
"No messages to get"	The requested message does not exist.
"Unknown state"	The state given in the request was not recognized.
"No s_umid supplied"	The ACK/NAK request cannot succeed without a s_umid.
"No s_umid to ACK"	There is no outstanding message requiring acknowledgment.
"No s_umid to NAK"	There is no outstanding message requiring acknowledgment.
"Incorrect s_umid, pending s_umid supplied"	<ul> <li>The s_umid given in the request is not the s_umid requiring acknowledgment. This response contains two additional arguments.</li> <li>s_umid: The s_umid of the message that is pending acknowledgment.</li> <li>instance: The instance number of that message.</li> </ul>
"Failed to route message:"	After the message was added to the routing point, it couldn't be "routed on" in SWIFT terminology. The reason for this failure is included in the response Content.
"Failed to reserve message:"	<ul> <li>The message specified could not be reserved. This response contains additional arguments on why the request failed.</li> <li>s_umid: The s_umid of the message that could not be acknowledged.</li> <li>instance: The instance number of the message that could not be acknowledged.</li> </ul>
"Failed to get message:"	SEWS was not able to retrieve the message. The response content provides text that explains the reason for the error.

Content	Meaning
"Failed to get message, then failed to unreserve it:"	SEWS reserved the message, but could not retrieve it, and then could not unreserve the same message when recovering. Includes text further detailing the problem in the content.
"Failed to count/list instances"	SWIFTAlliance couldn't perform the operation. The human readable text - Content - includes the ADK error string that details the problem.
"Failed to add message:"	<ul> <li>The message could not be added to the SWIFT routing point. The response content provides additional detail on why the procedure failed.</li> <li>When this error occurs, there are two additional arguments in the response.</li> <li>offset: The character offset into the text message, that caused the error.</li> <li>reason: Text giving further details about the error.</li> </ul>
"Unknown message type"	<ul> <li>The type argument contained an unexpected value. The response also contains a type argument:</li> <li>type: The type value given in the initial request that was not understood.</li> </ul>
"Message type not supplied"	No type argument was supplied with the initial GET request.
"Pending ACK for another message"	No more messages can be retrieved until the last message is acknowledged. Responses with this code include another argument "pending", which lists the s_umid of the pending message.
"Invalid routing point name supplied"	The routing point name given in the request was not recognized.
"No authorization"	Authentication was not possible.
"Too many clients connected"	SEWS cannot take more than one client per direction.

Content	Meaning
"Cannot translate SWIFT message to ADK"	The SWIFT message supplied in the request cannot be understood by SEWS. Perhaps there is a Signature error in the message. More exact details can be found in the SWIFTAlliance logs.
"Cannot translate TELEX message to ADK"	The Telex message supplied in the request cannot be understood by SEWS. Perhaps there is a Signature error in the message. More exact details can be found in the SWIFTAlliance logs.

# Index

# A

ADK - see Alliance Developer Toolkit Alliance Developer Toolkit (ADK) 7 authentication and authorization 17 Automated integration 7

### C

configuring enable authentication and authorization 17 enable SSL 17 host name 17 SeeBeyond JMS IQ Manager 17 server port 17 server SSL port 17 SWIFT ADK eWay 16 configuring SEWS 26

### E

eGate API Kit 7 Enable authentication and authorization enable SSL **17**, eWay Installation

### Η

host name **17**, **18** 

### 

implementation 20 installation 14 sar files 14 installing SEWS on Unix 23 SEWS on Windows 21 installing the SEWS eWay component 20 IP Address or host name 27

### L

Listening TQ Name 27 Logical Host requirements 13

### Μ

message flow SWIFT ADK eWay 8 Message Server Login 29 Message Type 30 QUEUE 30 TOPIC 30

## Ν

network parameters IP Address or host name 27 listening TQ name name 27 message server login name 27 message type 27 sending TQ name 27 TCP port number 27 use SSL Flag 27

## 0

```
operating systems
supported 12
overview 20, 43
SWIFT 6
SWIFT ADK eWay 7
```

### P

properties, eWay Project Explorer 17

# Q

Queues 36

### R

routing point SEWS\_from\_egate 28 Routing Points 36 routing points 7, 13

### S

sample projects 43 components 43 locating & importing 43 running 44 SWIFT\_ADK\_Sample\_BPEL 44 SWIFT\_ADK\_Sample\_JCE 43 Secured Sockets Layer 33 Security 7

SeeBeyond JMS IQ Manager 7 Sending TQ Name 31 SEWS\_to\_egate 31 server port 17, 18 server SSL port 17, 19 setting eWay properties Environment Explorer 16 setting up test environment 36 SEWS eWay component 20 SEWS Executable 7 SSL 33 Starting the SEWS Component 35 supported operating systems 12 Swift ADK API 7 SWIFT ADK eWay message flow 8 SWIFT ADK eWay Overview 7 SWIFT Network 7 SWIFT Overview 6 SWIFT ADK Sample BPEL 44 SWIFT\_ADK\_Sample\_JCE 43 SWIFTAlliance 7 SWIFTAlliance Access Control window 26 SWIFTAlliance parameters 27 synchronous (request/reply) 7 synchronous (send only) 7 System Management Window 26 system requirements 12 environment 13 Logical Host 13 Systems Management window network parameters 27

### Τ

TCP Port Number 32 test environment 36 Translation 7

### U

Use SSL Flag 33