Application

Using SWIFTNet

Standards Library Version 5.2

Sterling Commerce
An IBM Company

Contents

SWIFTNet with Application

Overview

Application supports the use of Society for Worldwide Interbank Financial Telecommunications (SWIFTNet), a standard for the financial industry from SWIFT™ that enables store-and-forward financial messaging through the InterAct file protocol.

The SWIFTNet standards data dictionary is optionally installed from the **Deployment > Standards** page. It supports all SWIFT Standards Release messages loaded in the standards database. The SWIFTNet standards data dictionary also contains the special exception and code word validations, and the codes words and qualifiers necessary for the validation of the ISO 15022 messages (500 series). This information is used to automatically generate the translator_swift.properties.in file, which is used by the translator to perform the validations.

Note: You can create a map for all SWIFTNet Standards Release messages through the Map Editor. All messages are validated by Application for syntax (that is, field types, field lengths, and so forth). All SWIFT message are validated for syntax *and* semantics.

The following messages are supported with both inbound and outbound syntax validation and semantic validation (that is, validating the message rules) of the messages between Application and SWIFTNet:

✦ All SWIFTNet 2005 message types

✦ All SWIFTNet 2006 message types

✦ All SWIFTNet 2007 message types

✦ SWIFTNet FUNDS (version 1.0, 3.0, and 4.0) and FUNDS Rulebook Version 4.0

✦ SWIFTNet Cash Reporting (version 3.0 and 3.1) and the Cash Reporting Rulebook Version 3.1

✦ SWIFTNet Exceptions & Investigations (version 1.0 and 1.1)

✦ SWIFTNet Trade Services (version 1.0)

Additionally, Application allows you to create maps to support the translation of market practices, and provides the following two Market Practices:

✦ Germany: MT515: Trade Confirmation (Broker to Asset Manager)

✦ United States: MT536: ISITC-IOA: Statement of Transactions

For Market Practices, the SWIFT standard validations are performed unless the Market Practice has customized a particular validation (for example, restricting a code word validation list). Market Practice code word and qualifier validations are maintained in property files separate from the SWIFT standard property files. By keeping the Market Practices you create separate, we can deliver updates to the SWIFT standard without overwriting your custom-implemented Market Practices. Similarly, semantic validation rules for Market Practices are stored in an extended rule library separate from the library that implements the SWIFT standard semantic validation rules. Please note that the standard property files and semantic validation rules are used when the implemented Market Practice does not override them.

In addition, support for SWIFTNet requires you to create inbound and outbound SWIFTNet envelopes. You must also create a business process or processes to order the flow of Application activities so you can accomplish your business objectives with SWIFTNet.

For more information on the SWIFTNet standard, access this web site:

http://www.swift.com

This table describes how Application supports SWIFTNet:

Functionality	Enhancements	Action You Need to Take to Implement SWIFTNet
SWIFTNet Standards Data dictionary	◆ Provides all validation for field tags and structures for SWIFT messages. ◆ References a SWIFT extended rule library that is automatically checked in so you can use it with your maps. ◆ SWIFT_FINMessageTypes code list, supplied as part of the install, which contains a list of valid SWIFT message types. ◆ If you purchase a Financial Services license, you have access to XML standards once you download them.	◆ Populate four external codes lists: SWIFT_Addresses, SWIFT_BaseAddresses (only needs to be populated when address verification is enabled), SWIFT_Currencies, and SWIFT_Countries. ◆ If you want to use BIC+ Validation, you must also populate the SWIFT_ClearingCodes code list. See *SWIFTNet Envelopes* on page 14. ◆ Install the SWIFTNet standards data dictionary. **Note:** The data dictionary is optionally installed from the Deployment > Standards page. The SWIFT extended rules library is automatically installed when you download the Map Editor.
SWIFTNet Market Practices	The following two market practices are available in Application: ◆ Germany: MT515: Trade Confirmation (Broker to Asset Manager) ◆ United States: MT536: ISITC-IOA: Statement of Transactions You can create a new message type by customizing a standard SWIFT message type map according to the specifications of the Market Practice. The customized map can be imported into the SWIFTNet Market Practices data dictionary through the SWIFTRuleImporter utility.	◆ Install the SWIFTNet standards data dictionary. The data dictionary is optionally installed from the Deployment > Standards page. The SWIFT extended rules library is automatically installed when you download the Map Editor. ◆ You can create a new message type by implementing the additional restrictions related to the desired Market Practice.

Functionality	Enhancements	Action You Need to Take to Implement SWIFTNet
Services	◆ Generic Envelope service envelopes SWIFT messages. **Note:** The Generic envelope service assumes that SWIFT expects a starting CRLF (carriage return/line feed) but not an ending CRLF. ◆ Generic Deenvelope service deenvelopes SWIFT messages. **Note:** The Generic Deenvelope service assumes that SWIFT expects a starting CRLF (carriage return/line feed) but not an ending CRLF. ◆ EDI Deenvelope service accepts SWIFT messages. ◆ SWIFTNet Server Adapter. ◆ SWIFTNet Client Service. ◆ SWIFTNet Reconciliation Service. ◆ SWIFTNet HTTP Server Adapter. ◆ An application, SWIFTNet MEFG Server, supports FileAct and InterAct processing, as well as failover processing.	◆ Install the Application SWIFTNet MEFG Server on a machine running either the Sun Solaris 5.10, Windows Server 2003 (Standard or Enterprise Edition), or AIX 5.3 operating system. ◆ Configure the SWIFTNet Client Service. ◆ Configure the SWIFTNet Server adapter.
Envelopes	◆ Inbound SWIFT envelope wizard to implement inbound SWIFTNet. ◆ Outbound SWIFT envelope wizard to implement outbound SWIFTNet. ◆ The SWIFT_FINMessageTypes code list is automatically installed with the Application. This code list contains a list of valid message types.	◆ Create the appropriate SWIFT envelopes for each message type you are sending and receiving. ◆ You need to populate four code lists to use in conjunction with the SWIFT_FINMessageTypes code list to perform SWIFTNet validations. ◆ Configure the EDI Encoder service for use with the outbound SWIFT messages.

Functionality	Enhancements	Action You Need to Take to Implement SWIFTNet
Predefined business processes	◆ SWIFTDeenvelope business process ◆ SWIFTEnvelope business process ◆ SWIFTNetClient business process ◆ SWIFTNetClientFA business process ◆ handleSWIFTNetServerSnFRequest ◆ handleSWIFTNetInboundCorrelation ◆ handleSWIFTNetOutboundCorrelation ◆ handleSWIFTNetServerFADelNotif ◆ handleSWIFTNetServerFAEvent ◆ handleSWIFTNetServerFARequest ◆ handleSWIFTNetServerFASnFDelNotif ◆ handleSWIFTNetServerFASnFRequest ◆ handleSWIFTNetServerRequest ◆ handleSWIFTNetServerSnFDelNotif ◆ handleSWIFTNetServerFASnFEvent ◆ handleSWIFTNetSnFInboundCorrelation ◆ handleSWIFTNetSnFOutboundCorrelation ◆ SWIFTMessageEntryEnvelope	The business processes that are related to the SWIFTNet workflow must have the Document Tracking option enabled when you check in or edit the business process. Additionally, you need to configure other parameters in the SWIFTNetClient business process to support SWIFTNet.

Functionality	Enhancements	Action You Need to Take to Implement SWIFTNet

Map Editor	◆ Map Editor wizard enables you to generate a file layout using the MT or MX message that you select. Included in the SWIFTNet map are groups, records, composites, and fields that are defined by SWIFT.
	
	◆ Properties dialog boxes that enable you to define and modify SWIFTNet map components.
	
	◆ Autolink function automatically creates links between input and output fields that have the same name or business name. This function can be used with any data format.
	
	◆ Extended Rules Library function (used with SWIFTNet and any other data format) contains a list of rules in a separate file outside of the Map Editor source. Map Editor stores the name of the library in its source file, so when you load a map the library is also loaded and compiled. This enables you to create a library of extended rules and then add it to any other map, so you do not have to recreate those extended rules after the first time.
	
	◆ Extended rules are used to validate SWIFTNet messages.
	
	◆ SWIFTNet MX validations are performed and generated maps define the parameters used to look up validations that are defined in a properties file.
	
	◆ The extended rule libraries that are loaded automatically when you download SWIFT standards are:
	  ◆ SWIFT_2006.erl (2006 MT and MX messages)
	  ◆ SWIFT_2007.erl (2006 MT and MX messages)
	  ◆ SWIFTMX.erl (Common Rule Library)
	  ◆ SWIFTMX_Funds_v4-0.erl for Funds 4.0
	  ◆ SWIFTMX_CashReporting_3-1.erl for CashReporting 3.1
		◆ Download Map Editor.
		
		◆ Create a map or maps to translate your SWIFTNet messages.
		
		◆ Place custom extended rule code inside a validation block.

Functionality	Enhancements	Action You Need to Take to Implement SWIFTNet
SWIFTNet Correlations	◆ Enables you to search for SWIFTNet messages using specific criteria.	◆ Run inbound and outbound data and then use the search interface to search for messages.
Document Tracking	◆ Support for monitoring SWIFTNet data flows. ◆ Support for tracking SWIFTNet messages within Application. The Application tracking and correlation functionality automatically tracks the SWIFTNet messages the same manner as the other standards are tracked. ◆ Support for document repair and resend.	◆ Use the data flow interface to track inbound and outbound SWIFTNet processes. ◆ All the business processes that are related to the SWIFTNet workflow (SWIFTNetClient, handleSWIFTNetServerRequest, and handleSWIFTNetServerSnfRequest) must have the Document Tracking option enabled when you check in or edit the business processes. ◆ Access messages with Ready To Edit status, repair, and resend them.
SWIFTNet Routing Rule	◆ Interface to support creation, modification, and deletion of SWIFTNet routing rules. ◆ The criteria specified in the SWIFTNet Routing Rule enables a user to set a specific business process to process the incoming request from SWIFTNet.	◆ Create a business process or modify the SWIFTNetClient business process. ◆ Create a SWIFTNet routing rule that is linked to the business process you created.
SWIFT Message Editor	◆ Interface to enable a user to correct a SWIFTNet MT or MX message that was returned due to an error either in transmission or translation. ◆ Support for validating messages.	◆ Assign permissions for two different roles: the Editor and the Reviewer and give appropriate permission sets to each role. ◆ Correct appropriate messages and notify the appropriate user that activity is required. ◆ Review change history for messages.
SWIFT Message Entry Workstation	Interface to enable manual message entry using a structured message editor.	◆ Assign permissions for the three different roles (Creator, Verifier, Tracker). ◆ Create, verify, and track new SWIFT messages created through this interface.

Prerequisite Knowledge

The audience using this software should be familiar with Application, the SWIFTNet standard, and using InterAct and/or FileAct.

Note: This documentation is not intended to explain the SWIFTNet standard.

Using SWIFTNet with Application

To use SWIFTNet with Application, you must complete the following tasks:

1. Download the SWIFTNet standards data dictionary. See *Downloading and Installing the SWIFTNet Standards Data Dictionary* on page 13.

2. Create inbound and outbound SWIFTNet envelopes. See *SWIFTNet Envelopes* on page 14.

3. Create the necessary code lists and maintain them as needed. See *Maintaining the External Code Lists* on page 16.

4. Enable the document tracking option for all the business processes that are that are related to the SWIFTNet workflow. See *Document Tracking for SWIFTNet Transport* on page 164 and *SWIFT Business Processes* on page 59.

5. Configure the EDI Encoder service for use with outbound SWIFTNet messages. See *Configuring the EDI Encoder Service for SWIFTNet Outbound Messages* on page 57.

6. Create the appropriate SWIFTNet maps. See *Creating SWIFTNet Maps* on page 101.

7. Use SWIFT Alliance Access (SAA) for SWIFT support.

8. Use the WebSphere MQ adapter to communicate with SAA.

Using InterAct with Application

To use InterAct with Application, you must complete the following tasks:

1. Configure the appropriate service parameters, including failover processing configuration. See *SWIFTNet Server Adapter* and *SWIFTNet Client Service*.

2. Configure the Application SWIFTNet MEFG Server for InterAct processing. See *SWIFTNet MEFG Server* on page 147.

Using FileAct with Application

To use FileAct with Application, you must complete the following tasks:

1. Configure the appropriate service parameters. See *SWIFTNet Server Adapter* and *SWIFTNet Client Service*.

2. Configure the Application SWIFTNet MEFG Server for FileAct processing. See *SWIFTNet MEFG Server* on page 147.

Downloading and Installing the SWIFTNet Standards Data Dictionary

Before you install the SWIFTNet standards data dictionary (standards database) on your desktop, consider these guidelines:

✦ Download the Map Editor.

✦ For the most current version of the SWIFTNet standards, contact Sterling Commerce Customer Support.

✦ Be sure your desktop meets the Windows Client requirements listed in the Application *System Requirements*.

To download and install the standards database:

1. From the Application **Deployment** menu, select **Standards**.

2. In the Download and Install section next to Download SWIFT Standards, click **Go!**

3. In the **File Download** dialog box, select a download option, then click **OK**.

    ◆ If you choose to run the file click **Run** and the operating system downloads the files immediately.

    ◆ If you choose to save the file, the operating system prompts you to save the file. Browse to the location where you want to download the file and click **OK**. If you want to continue installing, run the file you just saved from the location you specified.

4. In the Security Warning page, select **Always trust content from Sterling Commerce (Mid America), Inc.** if you do not want to see similar security messages in the future when you download software from Sterling Commerce. Click **Yes**.

5. In the Welcome window, click **Next**.

6. In the Choose Destination Location window, select where you want to install the standards database:

    ◆ If you accept the default location, click **Next**.

    ◆ If you want to specify a different location, click **Browse**, specify the path to the folder, click **OK**, and click **Next**.

    If you specify a folder name that does not exist, Application displays a message asking if you want to create that folder.

7. In the Select Components window, verify that **SWIFTStandardDatabase** is selected (if you want to use the preloaded SWIFT market practices, also ensure that **SWIFTMarketPractice** is selected) and click **Next**.

    The download wizard installs the standards database.

8. In the Setup Complete window, click **Finish**.

SWIFTNet Envelopes

Overview

A *document envelope* consists of control information that enables organizations to effectively exchange messages. This information is added in headers and trailers to messages. Document envelopes are specific to the message protocol used. Creating document envelopes is necessary to use SWIFTNet with your trading partners.

SWIFTNet has only one level of envelope, which you must modify appropriately to reflect your information and your trading partner's information. Envelopes specify whether the message is inbound or outbound:

✦ The **Inbound SWIFT envelope** identifies messages that are received by Gentran Integration Suite so they can be properly routed. Inbound envelopes also give you the option to translate messages when you choose to check messages for compliance. By choosing to translate messages from within the envelope, you can reduce message processing time because you do not need to specify a separate Translation service step in the business process. You need to create an Inbound SWIFT envelope to configure deenveloping information for MT or MX messages. See *Inbound SWIFT envelope* on page 19.

✦ The **Outbound SWIFT envelope** identifies messages so they can be sent to and received by trading partners. You need to configure an Outbound SWIFT envelope to configure enveloping information for MT or MX messages. See *Outbound SWIFT envelope* on page 36.

When you envelope an outbound SWIFTNet message, the SWIFTNet header and trailer are created. For an inbound message, the envelope contains the header information (the trailer information is a summary appended to the SWIFTNet data). The header types are MX or MT messages.

MT and MX messages are included within the Body element. MT messages are base-64 encoded and MX messages are XML.

Application also enables you to use SWIFT XML Format 2, the envelope supported by SWIFT Alliance Access 6.

As part of SWIFTNet enveloping, Application uses code lists to validate the data. Application uses code pairs in code lists to identify items in transactions between two or more trading partners. A trading partner code list consists of one or many pairs of code values containing a sender code and a receiver code. Each code pair has one description and up to four additional codes relating to the pair. Code lists are dynamic and are stored in a database.

The SWIFT_FINMessageTypes code list is automatically installed with Application. This code list contains a list of valid SWIFT message types. The three-digit message number is entered for sender code and receiver code, and the description is set to **SWIFT Message Type**. However, you need to populate four additional code lists to perform SWIFTNet validations:

✦ **SWIFT_Addresses**—used to check the sender and receiver IDs within the message. This code list is shared with message authentication. The code list used is the same code list that is used for message validations. You will type the address in the Sender Code, Receiver Code, and Description parameters, and use the Text1 parameter to indicate the subtype of the address. The SWIFT_Addresses and SWIFT_BaseAddresses code lists are used to differentiate between bad base addresses and bad branch

codes when necessary. The SWIFT_Addresses code list is also used for verification of those addresses that are contained within the body of a message.

✦ **SWIFT_BaseAddresses**—this is a list of the 8-character address (the BIC minus the branch code) that are valid as part of a sender address when generating a message. You will type the eight-digit code in the Sender Code, Receiver Code, and Description parameters. The SWIFT_Addresses and SWIFT_BaseAddresses code lists are used to differentiate between bad base addresses and bad branch codes when necessary.

Note: You only need to populate the SWIFT_BaseAddresses code list when you have enabled address verification. See *Enabling and Disabling Address Verification* on page 16.

✦ **SWIFT_Currencies**—this is a list of the valid currencies that can be used in a SWIFT message. You will use the Text1 parameter to indicate the maximum number of digits after the decimal point that the currency supports.

✦ **SWIFT_Countries**—this is a list of the valid countries that can be used as part of the address when generating a SWIFT message.

✦ **SWIFT_ClearingCodes**—this is a list of the clearing codes used when validating a message. You must populate this list if you want to use BIC+ validation.

See *Maintaining the External Code Lists* on page 16.

SWIFT MT validations are also applied to SWIFT MX messages for currencies, country codes, BIC/BEI addresses, and International Bank Account Number checksum validation (IBAN). Application allows you to define codes lists for currencies, countries, and BIC or BEI addresses (which are validated against the SWIFT_Addresses code list). IBAN data contains a country code that is validated against the SWIFT_Countries code list in Application, and additional IBAN validation is handled internally by the translator.

The validation of the SWIFT special functions <CUR>, <SWIFTBIC>, and <NON-SWIFTBIC> use these code lists. You must update and maintain these codes lists, as necessary.

For outbound SWIFTNet messages, you also need to configure the EDI Encoder service to include the proper values for the following parameters:

✦ AccepterLookupAlias

✦ ReceiverID

✦ SenderID

✦ ReceiverIDQual

✦ SenderIDQual

See *Configuring the EDI Encoder Service for SWIFTNet Outbound Messages* on page 57.

Note: When editing a SWIFT envelope, if you skip wizard pages by clicking **Save** or clicking on a later step, the final confirmation page will display unexpected values. Also, if you skip page in the envelope wizard and then use the **Back** button, incorrect pages may be displayed. If you experience this issue, save from the confirmation screen without using the **Back** button and the envelope is saved correctly. However, if you use the **Back** button, an unexpected page may be displayed (that is, a page that would not normally be displayed based on the envelope values), and the page may hang or cause you to enter an improper value. If this occurs, cancel out of the envelope wizard, and start editing the SWIFT envelope again.

Enabling and Disabling Address Verification

Application allows you to enable or disable address verification. Address verification is performed using the SWIFT_Addresses and SWIFT_BaseAddresses code lists. See *Maintaining the External Code Lists* on page 16 for more information on creating these code lists.

Configuring Inbound Address Verification

The **enveloping.verify_addresses_while_deenveloping.SWIFT_FIN_INBOUND** property enables and disables inbound address verification.

To enable *inbound* address verification, complete these steps:

1. Access the *install_dir*/**properties/enveloping.properties.in** file, and change the line **enveloping.verify_addresses_while_deenveloping.SWIFT_FIN_INBOUND** to **TRUE**, as noted below:

```
enveloping.verify_addresses_while_deenveloping.SWIFT_FIN_INBOUND=True
```

Note: If you want to then disable inbound address verification, you can do so by accessing the *install_dir*/**properties/envelope.properties.in** file, and change the line **enveloping.verify_addresses_while_deenveloping.SWIFT_FIN_INBOUND=FALSE**.

2. Save and close the **enveloping.properties.in** file.

3. Stop Application.

4. Run the setupfiles script using one of the following steps:

   ◆ (UNIX or Linux) - From the *install_dir*/bin directory, run the **setupfiles.sh** command.

   ◆ (Windows) - From the *install_dir*\bin directory, run the **setupfiles.cmd** command.

5. Start Application.

Configuring Outbound Address Verification

To enable or disable *outbound* address verification, use the Outbound SWIFT envelope parameter **Validate Sender and Receiver**. This parameter allows you to enable (by choosing **Yes**) or disable (by choosing **No**) address verification. The default is **No** (sender and receiver verification is disabled). See *Outbound SWIFT envelope* on page 36 for more information.

Maintaining the External Code Lists

All necessary code lists are automatically installed but except for the Message Type code lists, they are not automatically populated with codes. Therefore, you need to populate the following five external code lists for use in conjunction with the SWIFT_MessageTypes code list (all code lists are automatically installed with Application—the SWIFT_MessageTypes code list already contains a list of valid message types) to perform SWIFTNet validations.

The five external code lists for which you need to populate SenderID, ReceiverID, Description, and occasionally Text1 are:

✦ SWIFT_Addresses
✦ SWIFT_BaseAddresses

- ✦ SWIFT_Currencies
- ✦ SWIFT_Countries
- ✦ SWIFT_ClearingCodes

You populate these code lists automatically by using the HIPAA codelist conversion map for countries to populate it. However, prior to using the HIPAA codelist conversion map, you need to modify the import file created using the conversion map (the import file is used to populate the codelist). To modify the import file you need to change the codelist name from xxx to SWIFT_xxx (for example, change ClearingCodes to **SWIFT_ClearingCodes**).

To populate the each code list automatically:

1. From a command line, go to the **tp_import** directory.

2. Type the following command to start the conversion and import process, where <map name> is the name of the map to use during translation (without the file extension) and <code list path and filename> is the fully qualified name of the code list to translate, including filename extension, if any:

   - ◆ If you are using Windows, **hipaaconvert.cmd [-import] <map name> <code list path and filename>**

   - ◆ If you are using UNIX, **hipaaconvert.sh [-import] <map name> <code list path and filename>**

   Do not specify the file extension for the map name when importing a code list—just indicate the base name of the map.

   The [-import] parameter is optional. You can convert the code list file without importing it. If you do not use the [-import] parameter during conversion, you can import the resulting XML file into Application using the import utility.

   The input files and maps for each code list are as follows:

Code List	Map	Input File and Directory
SWIFT_Addresses	SWIFTFIFIleToSWIFT_Addresses	FI file from BIC directory
SWIFT_BaseAddresses	SWIFTFIFIleToSWIFT_BaseAddresses	FI file from BIC directory
SWIFT_ClearingCodes	SWIFTFIFIleToSWIFT_ClearingCodes	FI file from BIC+ directory
SWIFT_Countries	SWIFTCTFileToSWIFT_Countries	CT file from BIC directory
SWIFT_Currencies	SWIFTCUFileToSWIFT_Currencies	CU file from BIC directory

3. Once the utility completes, a translation report (hipaaconvert.rpt) and an input file (hipaaconvert.xml) are created. If no translation errors are reported, the code list was successfully generated (and imported if you used the [-import] parameter). A code list will not be imported if there are translation errors.

4. To modify the import file, change the codelist name from xxx to **SWIFT_xxx** so it can be used for SWIFTNet messages.

Disabling Automatic BIC+ Validation for the SWIFT_ClearingCodes Code List

The SWIFT_ClearingCodes code list is the only code list that uses BIC+ validation. If you want to disable the automatic BIC+ validation, access the **translator_swift_2006.properties** file and change it to the following:

```
fieldTags.BICPlus.102_not_STP=53A,52C57A,57C

fieldTags.BICPlus.102_STP=52A,57A

fieldTags.BICPlus.103_not_STP=52A,52D56A,56C,57A,57C,57D

fieldTags.BICPlus.103_STP=52A,56A,57A
```

Creating Envelopes

Inbound envelopes define expected header and trailer information for inbound messages. This information helps Gentran Integration Suite route and process the messages. Outbound envelopes specify information about messages that enables them to be sent to and received by trading partners, and they gather and provide the appropriate data used to create the header.

To create an envelope:

1. From the Admin Console, select **Trading Partner > Document Envelopes > Envelopes**.

2. Under Create, next to New Envelope, click **Go!**

3. On the Envelope Standards page, select **SWIFT** and click **Next**.

4. Select the level of envelope you want to create, inbound or outbound, and click **Next**.

5. On the Base Envelope page, do you want this envelope to inherit properties from a base envelope (if available)?

   ◆ If Yes, select a base envelope and click **Next**.

   ◆ If No (you want to create a new envelope), select **Not Applicable** and click **Next**.

6. On the Name page, type a unique name for the envelope, and a description or comments, then click **Next**.

7. Complete the properties for the envelope as necessary and click **Next** after each page until you reach the confirm page. Required fields are highlighted in blue. See *Inbound SWIFT envelope* on page 19 or *Outbound SWIFT envelope* on page 36.

8. Click **Finish** to add the envelope.

Using Base Envelopes

A *base envelope* is a regular envelope that you use as a starting point to create a new envelope. The base envelope maintains a link to the envelope that inherited its properties. If you modify the base envelope, all related envelopes (those that inherited the base envelope properties) are also changed.

When you create an envelope using a base envelope, everything in the new envelope is the same as in the base envelope, except the envelope name, description, and parameters such as unique identification numbers. If you plan to create many envelopes using base envelopes, do not use the base envelopes in production. You should also be sure to note the envelopes that are related to the base envelopes.

To use a base envelope:

1. Create the base envelope, using *Creating Envelopes* on page 18.
2. Specify the name and description of a new envelope.
3. Identify the base envelope that the new envelope uses.

Inbound SWIFT envelope

You only need to create an Inbound SWIFT envelope if you are receiving inbound SWIFTNet messages. However, if you are receiving inbound SWIFTNet messages, you need to create a separate Inbound SWIFT envelope for *each* SWIFTNet message type you will be receiving. The following table describes Inbound SWIFT envelope properties:

Note: An (*) asterisk indicates that a wildcard value can be used with that parameter (for mandatory fields, the wildcard value is an (*) asterisk and for optional fields, the wildcard value is leaving the field blank). For Inbound envelopes, a wildcard value in the envelope matches any value in the input document, while an empty value in the envelope matches only an empty value in the input document. For Outbound envelopes, a wildcard value is equivalent to an empty value in the envelope.

Field or Check Box	Description
* Sender ID	Coded identifier of the supplier number or data sender. Valid value is eight standard characters for BIC 8. Optional.
	Note: This parameter enables you to type in a new ID or pick an ID that has already been used. When you start typing an ID, Application returns all matching IDs existing in the system and provide a combo-box from which you can select an ID by double-clicking it. There must be Sender ID Codes in the system for autocomplete to find matches and display a selection list.
	Note: It is possible to leave the Sender ID parameter blank under some circumstances. If you leave this parameter blank, it will cause errors during the enveloping process. You must type a valid value in the Sender ID parameter.
Sender ID Type	Type of sender identifier. Valid values are * (wildcard), BIC8 (default), Nickname, Distinguished Name. Required.
	Note: The full address for BIC8 is a BIC12, which includes the one-character logical terminal (which is specified for the sender and is always "X" for receiver), and the 3-character branch code. These are combined to form the BIC12.

Field or Check Box	Description
* Receiver ID	Coded identifier of the customer number or data source number. Valid value is eight standard characters for BIC 8. Optional.
	Note: This parameter enables you to type in a new ID or pick an ID that has already been used. When you start typing an ID, Application returns all matching IDs existing in the system and provide a combo-box from which you can select an ID by double-clicking it. There must be Receiver ID Codes in the system for autocomplete to find matches and display a selection list.
	Note: It is possible to leave the Receiver ID parameter blank under some circumstances. If you leave this parameter blank, it will cause errors during the enveloping process. You must type a valid value in the Receiver ID parameter.
Receiver ID Type	Type of receiver identifier. Valid values are * (wildcard), BIC8 (default), Nickname, Distinguished Name. Required.
	Note: The full address for BIC8 is a BIC12, which includes the one-character logical terminal (which is specified for the sender and is always "X" for receiver), and the 3-character branch code. These are combined to form the BIC12.
Envelope Format	The format of the envelope. Valid values are * (wildcard), FIN (default), and SAA XML Format 2. Required.
Message Category	The category of the message. Required. Valid values are: ◆ ACK/NAK (FIN only) ◆ Delivery Notification (XML Format 2) ◆ Delivery Report (XML Format 2) ◆ History Report (XML Format 2) ◆ Message (FIN or XML Format 2) (this is the default) ◆ Message Status (XML Format 2) ◆ Session Status (XML Format 2) ◆ Transmission Report (XML Format 2)
Enforce Message Size Limit	Specifies that Application will check the size of the message, and error out if the message exceeds it. If you select this check box, Application gives you the option to set the Maximum Message Size (which defaults to 10,000, the standard limit for SWIFT messages). Valid values are Yes (default) and No. Required.
Validate DataPDU Signature	Whether to validate the protocol data unit (PDU) for a signature. Optional. Valid values are Yes or No (default).
	Note: Only displayed if Envelope Format is set to either wildcard or SAA XML Format 2 is selected. The signature is optional.
Key Part 1	First part of the validation signature.
Key Part 2	Second part of the validation signature.

Field or Check Box	Description
Maximum Message Size	The maximum size of message that is allowed. The default is 10,000. This parameter only accepts numeric input. Only displayed if you selected Yes for the **Enforce Message Size Limit** parameter. Required.
* Sender Branch Code	Three-character code that further refines the Sender ID. For example, if the SenderID is ROMANSFL, the branch code might be WAS to indicate the Washington branch. This parameter acts as the Sender ID qualifier for envelope matching. Required. **Note:** Only displayed if Sender ID Type is set to BIC8. Sender Branch Code is not used for SAA XML v2.0 and do not apply when Distinguished Name and Nickname are used.
* Receiver Branch Code	Three-character code that further refines the Receiver ID. For example, if the ReceiverID is ROMANSFL, the branch code might be WAS to indicate the Washington branch. This parameter acts as the Receiver ID qualifier for envelope matching. Required. **Note:** Only displayed if Receiver ID Type is set to BIC8. Only displayed if Receiver ID Type is set to BIC8. Receiver Brach Code is not used for SAA XML v2.0 and do not apply when Distinguished Name and Nickname are used.
MX Business Area	The MX business area. Valid values are * (wildcard), Funds, Trade Services, Exceptions and Investigations, and Cash Reporting. Required. **Note:** Only displayed if you set Message Format to MX.

Field or Check Box	Description
Message Type	Transaction message type as determined by the information type in the header of the message group, which includes the message, or determined by the information type in the transaction message. Required.
	Note: The message type for MX messages are the full 4-component string (for example, camt.003.001.02). You are first allowed to select a Message Format of wildcard, MT, and MX. If you select a wildcard, this Message Type parameter is not displayed because it is assumed to be wildcard as well (since there is no reason to wildcard the format and then select a specific message). If you select MT as the Message Format, you will be given a list of MT messages for this Message Type parameter. If you choose MX, you receive a list of the "categories" (SWIFTNet Funds, SWIFTNet Trade Services, SWIFTNet Exceptions and Investigations, SWIFTNet Cash Reporting) for this Message Type parameter. Once you select the subtype, you are presented with a list of the message types for that category, in the format camt.003.001.02 – Get Account Information. The envelope UI code will be modified to allow us to do this while still saving the value under a single envelope parameter.
	The lists of Message Types are generated from two property files. The first property file (ui_swift_message_types.properties) will contain the base lists of the types provided with Application. The second property file (ui_swift_message_types_customer.properties) is user-modifiable, and can also contain additional messages that you want to add. If you add Message Types to the ui_swift_message_types_customer.properties property file, modify it in the customer_overrides.properties file. You are not required to restart Application when you edit the ui_swift_message_types_customer.properties file.
Message Format	The format of the message. Valid values are * (wildcard), MT (default), and MX. Required.
APC/FIN	For an MT message using XML Format 2, this indicates whether it is a system (APC) message or a user message (FIN). Required. Valid values are * (wildcard—this is the default), APC, and FIN.
Closed User Group	A specific set of trading partners defined within the SWIFT network. Optional.
Validation Flag (tag 119)	Specifies how to validate the message. Required. Default is * (wildcard).
Reconcile Message User Reference against a control number	Whether to reconcile the Message User Reference (MUR) against a control number. Valid values are Yes (default) and No. The MUR is a message identifier separate from the one SWIFT assigns. Required.

Field or Check Box	Description
Use global control number	Whether to use a global control number. Required. Valid values are: ◆ Yes (default) ◆ Yes (and generate name from data) **Note:** If you select this option, you must select at least one parameter for Primary Name Format. ◆ No **Note:** Only displayed if you selected Yes for **Reconcile message User Reference (MUR) against a control number**.
Perform Duplicate Control Number (MUR) Checking	Determine control number/MUR duplications. Required. Valid values: ◆ Yes ◆ No (default) **Note:** Only displayed if you selected Yes for **Reconcile message User Reference (MUR) against a control number**.
Maximum age of Control Number History Records in days	Maximum days that Gentran Integration Suite should retain a history of control numbers to use for duplication determinations. Valid value is nine numerics. Optional. **Note:** Only displayed if you selected Yes for **Reconcile message User Reference (MUR) against a control number**.
Assign control number	Select a control number with this envelope. Optional. Displayed only if **Use Global Control Number** is set to Yes
Local control number	Select a local control number to associate with this envelope. Default is 1. Required. Displayed only if **Use Global Control Number** is set to No.

Field or Check Box	Description
Primary Name Format	Check boxes to instruct what information to include when generating a name for a primary global control number and finding the correct number to assign based on that name. Optional. Select from: ◆ Sender ID ◆ Sender Branch Code ◆ Receiver ID ◆ Receiver Branch Code ◆ Message Format (MT/MX) ◆ Message Type ◆ Validation Flag (MT only) ◆ Envelope Format (FIN/XML2) **Note:** If you are using a generated global control number (**Use global control number** is set to Yes (and generate name from data)), Application tries to generate and match the following control numbers: ◆ First, it tries to generate and match the control number in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the control number SWIFT_100_Inbound_FIN. ◆ Second, it tries to generate and match the First Backup Name. ◆ Third, it tries to generate and match the Second Backup Name. ◆ Fourth, if there is not an existing control number that matches the names generated in steps 1-3, a control number with the name assigned in the first step is created.

Field or Check Box	Description
First Backup Name Format	The first backup name format to use when generating a global control number. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional. Select from: ◆ Sender ID ◆ Sender Branch Code ◆ Receiver ID ◆ Receiver Branch Code ◆ Message Format (MT/MX) ◆ Message Type ◆ Validation Flag (MT only) ◆ Envelope Format (FIN/XML2) **Note:** If you are using a generated global control number (**Use global control number** is set to Yes (and generate name from data)), Application tries to generate and match the following control numbers: ◆ First, it tries to generate and match the control number in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the control number SWIFT_100_Inbound_FIN. ◆ Second, it tries to generate and match the First Backup Name. ◆ Third, it tries to generate and match the Second Backup Name. ◆ Fourth, if there is not an existing control number that matches the names generated in steps 1-3, a control number with the name assigned in the first step is created.

Field or Check Box	Description
Second Backup Name Format	The second backup name format to use when generating a global control number. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional. Select from: ◆ Sender ID ◆ Sender Branch Code ◆ Receiver ID ◆ Receiver Branch Code ◆ Message Format (MT/MX) ◆ Message Type ◆ Validation Flag (MT only) ◆ Envelope Format (FIN/XML2) **Note:** If you are using a generated global control number (**Use global control number** is set to Yes (and generate name from data)), Application tries to generate and match the following control numbers: ◆ First, it tries to generate and match the control number in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the control number SWIFT_100_Inbound_FIN. ◆ Second, it tries to generate and match the First Backup Name. ◆ Third, it tries to generate and match the Second Backup Name. ◆ Fourth, if there is not an existing control number that matches the names generated in steps 1-3, a control number with the name assigned in the first step is created.
Compliance check message	Check the message body for compliance. Required. Valid values are Yes (default) and No.
Keep translated document after compliance check	Whether to keep the translated document after the compliance check. Valid values are Yes and No (default). Required. **Note:** If you select Yes, the translated document replaces the primary document.
Map Name Mode	How to determine which map to use to perform a compliance check. Required. Valid values are: ◆ Specify (default) ◆ Generate from data (this is useful if you want to share across envelopes) **Note:** If you select this option, you must select at least one parameter for Primary Name Format.
Map Name	Which map to use to perform a compliance check (if **Compliance Check message** set to Yes and **Map Name Mode** set to Yes). The map must already be checked in to Application. Optional.

Field or Check Box	Description
Primary Name Format	Check boxes to instruct what information to include when generating and matching a name for the map. Optional. Select from: ◆ Sender ID ◆ Sender Branch Code ◆ Receiver ID ◆ Receiver Branch Code ◆ Message Format (MT/MX) ◆ Message Type ◆ Validation Flag (MT only) ◆ Envelope Format (FIN/XML2) **Note:** If you are using a generated map name (**Map Name Mode** is set to Generate from data), Application tries to generate and match the following maps: ◆ First, it tries to generate and match the map in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the map named SWIFT_100_Inbound_FIN. ◆ Second, it tries to generate and match the First Backup Name. ◆ Third, it tries to generate and match the Second Backup Name. ◆ Fourth, if there is not an existing map that matches the names generated in steps 1-3, an error is generated. **Note:** Only occurs if **Generate an error if no matching map is found** is set to Yes.

Field or Check Box	Description
First Backup Name Format	The first backup name format to use when generating a map name. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional.
	Select from:
	◆ Sender ID
	◆ Sender Branch Code
	◆ Receiver ID
	◆ Receiver Branch Code
	◆ Message Format (MT/MX)
	◆ Message Type
	◆ Validation Flag (MT only)
	◆ Envelope Format (FIN/XML2)
	Note: If you are using a generated map name (**Map Name Mode** is set to Generate from data), Application tries to generate and match the following maps:
	◆ First, it tries to generate and match the map in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the map named SWIFT_100_Inbound_FIN.
	◆ Second, it tries to generate and match the First Backup Name.
	◆ Third, it tries to generate and match the Second Backup Name.
	◆ Fourth, if there is not an existing map that matches the names generated in steps 1-3, an error is generated.
	Note: Only occurs if **Generate an error if no matching map is found** is set to Yes.

Field or Check Box	Description
Second Backup Name Format	The second backup name format to use when generating a map name. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional. Select from: ◆ Sender ID ◆ Sender Branch Code ◆ Receiver ID ◆ Receiver Branch Code ◆ Message Format (MT/MX) ◆ Message Type ◆ Validation Flag (MT only) ◆ Envelope Format (FIN/XML2) **Note:** If you are using a generated map name (**Map Name Mode** is set to Generate from data), Application tries to generate and match the following maps: ◆ First, it tries to generate and match the map in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the map named SWIFT_100_Inbound_FIN. ◆ Second, it tries to generate and match the First Backup Name. ◆ Third, it tries to generate and match the Second Backup Name. ◆ Fourth, if there is not an existing map that matches the names generated in steps 1-3, an error is generated. **Note:** Only occurs if **Generate an error if no matching map is found** is set to Yes.
Generate an error if no matching map is found	Specifies whether to generate an error if the Map Name selected is not found. Valid values are Yes (default) and No. **Note:** Only displayed if **Map Name Mode** is set to Generate from data.
Determine Error Business Process Name By	How to determine the business process name to use if there were errors in the compliance check. Required. Valid values are: ◆ Specifying a Business Process name (default) ◆ Generating a Business Process name from the data (this is useful if you want to share across envelopes) **Note:** If you select this option, you must select at least one parameter for Primary Name Format.

Field or Check Box	Description
Primary Name Format	Check boxes to instruct what information to include when generating and matching a name for the error business process name. Optional. Select from: ♦ Sender ID ♦ Sender Branch Code ♦ Receiver ID ♦ Receiver Branch Code ♦ Message Format (MT/MX) ♦ Message Type ♦ Validation Flag (MT only) ♦ Envelope Format (FIN/XML2) **Note:** If you are using a generated error business process name (**Determine Error Business Process Name By** is set to Generating a Business Process name from the data), Application tries to generate and match the following business process: ♦ First, it tries to generate and match the business process in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the business process SWIFT_100_Inbound_FIN. ♦ Second, it tries to generate and match the First Backup Name. ♦ Third, it tries to generate and match the Second Backup Name. ♦ Fourth, if there is not an existing business process that matches the names generated in steps 1-3, an error is generated. **Note:** Only occurs if **Generate an error if no generated business process name exists in the system** is set to Yes.

Field or Check Box	Description
First Backup Name Format	The first backup name format to use when generating an error business process name. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional.
	Select from:
	◆ Sender ID
	◆ Sender Branch Code
	◆ Receiver ID
	◆ Receiver Branch Code
	◆ Message Format (MT/MX)
	◆ Message Type
	◆ Validation Flag (MT only)
	◆ Envelope Format (FIN/XML2)
	Note: If you are using a generated error business process name (**Determine Error Business Process Name By** is set to Generating a Business Process name from the data), Application tries to generate and match the following business process:
	◆ First, it tries to generate and match the business process in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the business process SWIFT_100_Inbound_FIN.
	◆ Second, it tries to generate and match the First Backup Name.
	◆ Third, it tries to generate and match the Second Backup Name.
	◆ Fourth, if there is not an existing business process that matches the names generated in steps 1-3, an error is generated.
	Note: Only occurs if **Generate an error if no generated business process name exists in the system** is set to Yes.

Field or Check Box	Description
Second Backup Name Format	The second backup name format to use when generating an error business process name. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional.
	Select from:
	◆ Sender ID
	◆ Sender Branch Code
	◆ Receiver ID
	◆ Receiver Branch Code
	◆ Message Format (MT/MX)
	◆ Message Type
	◆ Validation Flag (MT only)
	◆ Envelope Format (FIN/XML2)
	Note: If you are using a generated error business process name (**Determine Error Business Process Name By** is set to Generating a Business Process name from the data), Application tries to generate and match the following business process:
	◆ First, it tries to generate and match the business process in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the business process SWIFT_100_Inbound_FIN.
	◆ Second, it tries to generate and match the First Backup Name.
	◆ Third, it tries to generate and match the Second Backup Name.
	◆ Fourth, if there is not an existing business process that matches the names generated in steps 1-3, an error is generated.
	Note: Only occurs if **Generate an error if no generated business process name exists in the system** is set to Yes.
Generate an error if no generated business process name exists in the system	Specify whether to generate an error if there is no match to the generated business process name in the system. Valid values are Yes (default) and No.
	Note: Only displayed if **Determine the Error Business Process By** is set to Generating the business process name from the data.
Business Process List	Select a previously created business process to associate with this envelope. Optional.
	Displayed only if **Determine Error Business Process Name By** is set to Specifying the business process.
Determine the Business Process By	How to determine the business process name to use if there were no errors in the compliance check. Required.Valid values are:
	◆ Specifying a business process
	◆ Generating the business process name from the data
	Note: If you select this option, you must select at least one parameter for Primary Name Format.

Field or Check Box	Description
Primary Name Format	Check boxes to instruct what information to include when generating and matching a name for the business process. Optional. Select from: ♦ Sender ID ♦ Sender Branch Code ♦ Receiver ID ♦ Receiver Branch Code ♦ Message Format (MT/MX) ♦ Message Type ♦ Validation Flag (MT only) ♦ Envelope Format (FIN/XML2) **Note:** If you are using a generated business process name (**Determine the Business Process By** is set to Generating the business process name from the data), Application tries to generate and match the following business process: ♦ First, it tries to generate and match the business process in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the business process SWIFT_100_Inbound_FIN. ♦ Second, it tries to generate and match the first Backup Name. ♦ Third, it tries to generate and match the second Backup Name. ♦ Fourth, if there is not an existing business process that matches the names generated in steps 1-3, an error is generated. **Note:** Only occurs if **Generate an error if no generated business process name exists in the system** is set to Yes.

Field or Check Box	Description
First Backup Name Format	The first backup name format to use when generating a business process name. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional.

Select from:

- Sender ID

- Sender Branch Code

- Receiver ID

- Receiver Branch Code

- Message Format (MT/MX)

- Message Type

- Validation Flag (MT only)

- Envelope Format (FIN/XML2)

Note: If you are using a generated business process name (**Determine Business Process Name By** is set to Generating a Business Process name from the data), Application tries to generate and match the following business process:

- First, it tries to generate and match the business process in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the business process SWIFT_100_Inbound_FIN.

- Second, it tries to generate and match the First Backup Name.

- Third, it tries to generate and match the Second Backup Name.

- Fourth, if there is not an existing business process that matches the names generated in steps 1-3, an error is generated.

Note: Only occurs if **Generate an error if no generated business process name exists in the system** is set to Yes.

Field or Check Box	Description
Second Backup Name Format	The second backup name format to use when generating a business process name. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional. Select from: ◆ Sender ID ◆ Sender Branch Code ◆ Receiver ID ◆ Receiver Branch Code ◆ Message Format (MT/MX) ◆ Message Type ◆ Validation Flag (MT only) ◆ Envelope Format (FIN/XML2) **Note:** If you are using a generated business process name (**Determine Error Business Process Name By** is set to Generating a Business Process name from the data), Application tries to generate and match the following business process: ◆ First, it tries to generate and match the business process in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the business process SWIFT_100_Inbound_FIN. ◆ Second, it tries to generate and match the First Backup Name. ◆ Third, it tries to generate and match the Second Backup Name. ◆ Fourth, if there is not an existing business process that matches the names generated in steps 1-3, an error is generated. **Note:** Only occurs if **Generate an error if no generated business process name exists in the system** is set to Yes.
Generate an error if no generated business process name exists in the system	Specify whether to generate an error if there is no match to the generated business process name in the system. Valid values are Yes (default) and No. **Note:** Only displayed if **Determine the Business Process By** is set to Generating the business process name from the data.
Business Process List	Select a previously created business process to associate with this envelope Optional. Displayed only if **Determine the Business Process Name** is set to Specifying the business process.
Extraction Options	Business process data extraction. Required. Valid values: ◆ Determined by business process (default) ◆ Extract to a file system directory ◆ Extract to a mailbox

Field or Check Box	Description
Data Extraction Directory	Directory for data extraction. Displayed only if Extraction Options set to **Extract to a file system directory**. Optional
Data Extraction Filename	Filename for data extraction. Displayed only if Extraction Options set to **Extract to a file system directory**. Optional.
Data Extraction Mailbox	Mailbox for data extraction. Displayed only if Extraction Options set to **Extract to a mailbox**. Optional.
Data Extraction Mailbox Message Name	Mailbox message name for data extraction. Displayed only if Extraction Options set to **Extract to a mailbox**. Optional.

Outbound SWIFT envelope

You only need to create an Outbound SWIFT envelope if you are sending outbound SWIFTNet messages. However, if you are sending outbound SWIFTNet messages, you need to create a separate Outbound SWIFT envelope for *each* SWIFTNet message type you will be sending. The following table describes Outbound SWIFT envelope properties:

Note: An (*) asterisk indicates that a wildcard value can be used with that parameter. For Inbound envelopes, a wildcard value in the envelope matches any value in the input document, while an empty value in the envelope matches only an empty value in the input document. For Outbound envelopes, a wildcard value is equivalent to an empty value in the envelope.

Field or Check Box	Description
Envelope Format	The format of the envelope. Required. Valid values are FIN (default) and SAA XML Format 2.
* Sender ID	Coded identifier of the supplier number or data sender. Valid value is eight standard characters for BIC 8. Optional.
	Note: This parameter enables you to type in a new ID or pick an ID that has already been used. When you start typing an ID, Application returns all matching IDs existing in the system and provide a combo-box from which you can select an ID by double-clicking it. There must be Sender ID Codes in the system for autocomplete to find matches and display a selection list.
	Note: It is possible to leave the Sender ID parameter blank under some circumstances. If you leave this parameter blank, it will cause errors during the enveloping process. You must type a valid value in the Sender ID parameter.
Sender ID Type	Type of sender identifier. Valid values are * (wildcard), BIC8 (default), and Distinguished Name. Required.
	Note: The full address for BIC8 is a BIC12, which includes the one-character logical terminal (which is specified for the sender and is always "X" for receiver), and the 3-character branch code. These are combined to form the BIC12.

Field or Check Box	Description
* Receiver ID	Coded identifier of the customer number or data source number. Valid value is eight standard characters for BIC 8. Optional.
	Note: This parameter enables you to type in a new ID or pick an ID that has already been used. When you start typing an ID, Application returns all matching IDs existing in the system and provide a combo-box from which you can select an ID by double-clicking it. There must be Receiver ID Codes in the system for autocomplete to find matches and display a selection list.
	Note: It is possible to leave the Receiver ID parameter blank under some circumstances. If you leave this parameter blank, it will cause errors during the enveloping process. You must type a valid value in the Receiver ID parameter.
Receiver ID Type	Type of receiver identifier. Valid values are * (wildcard), BIC8 (default), and Distinguished Name. Required.
	Note: The full address for BIC8 is a BIC12, which includes the one-character logical terminal (which is specified for the sender and is always "X" for receiver), and the 3-character branch code. These are combined to form the BIC12.
Accepter Lookup Alias	Identifying string used with the Sender ID and the Receiver ID to look up this envelope with the EDI Encoder service. This alias associates a message with the service it requires. Valid value must be at least one limited standard character. Required. Default is FIN.
Use Correlation Overrides	When to use correlation overrides (when a SWIFT Reviewer chooses not to validate a message on resend). Required. Valid values are: ◆ Never (default) ◆ Only when the envelope field has a wildcard value (*) ◆ Always ◆ Use the default specified in enveloping.properties
Batching Options	When the EDI Encoder is used to prepare multiple messages for enveloping, you can either choose to process each message individually or concatenate them into a single file, with the messages separated by a '$'. Required. Valid values are: ◆ Use FileAct batching ("$" between messages) (default) ◆ Do not batch messages
Enforce Message Size Limit	Specifies that Application will check the size of the message, and error out if the message exceeds it. If you select this check box, give you the option to set the Maximum Message Size (which defaults to 10,000, the standard limit for SWIFT messages). Valid values are Yes (default) and No. Required.

Field or Check Box	Description
Message User Reference (MT Message User Reference/MX Message Reference)	Indicates how to process the message user reference. The message user reference block is an optional section in the SWIFT envelope that the SWIFT network tracks for acknowledgement purposes. Valid values are: ◆ Don't include a message user reference (default) ◆ Use control number ◆ Use value from process data **Note:** You should only select Use value from process data if you are using immediate enveloping. With deferred enveloping, the process data values from the encoding step are not passed on to the enveloping step, so using a process data element will not work with deferred enveloping.
Message Format	The format of the message. Valid values are MT (default) and MX. Required.
Validate Sender and Receiver	Specify whether to validate the sender and receiver. Required. Valid values are Yes and No (default). **Note:** This parameter allows you to enable (Yes) or disable (No, which is the default) address verification. Address verification is performed using the SWIFT_Addresses and SWIFT_BaseAddresses code lists. See *Maintaining the External Code Lists* on page 16 for more information on creating these code lists.
MX Business Area	The MX business area. Valid values are * (wildcard), Funds, Trade Services, Exceptions and Investigations, and Cash Reporting. Required. **Note:** Only displayed if you set Message Format to MX.

Field or Check Box	Description
Message Type	Transaction message type as determined by the information type in the header of the message group, which includes the message, or determined by the information type in the transaction message. Required for both MT and MX messages.
	Note: The message type for MX messages are the full 4-component string (for example, camt.003.001.02). You are first allowed to select a Message Format of wildcard, MT, and MX. If you select a wildcard, this Message Type parameter is not displayed because it is assumed to be wildcard as well (since there is no reason to wildcard the format and then select a specific message). If you select MT as the Message Format, you will be given a list of MT messages for this Message Type parameter. If you choose MX, you receive a list of the "categories" (SWIFTNet Funds, SWIFTNet Trade Services, SWIFTNet Exceptions and Investigations, SWIFTNet Cash Reporting) for this Message Type parameter. Once you select the subtype, you are presented with a list of the message types for that category, in the format camt.003.001.02 – Get Account Information. The envelope UI code will be modified to allow us to do this while still saving the value under a single envelope parameter.
	The lists of Message Types are generated from two property files. The first property file (ui_swift_message_types.properties) will contain the base lists of the types provided with Application. The second property file (ui_swift_message_types_customer.properties) is user-modifiable, and can also contain additional messages that you want to add. If you add Message Types to the ui_swift_message_types_customer.properties property file, modify it in the customer_overrides.properties file. You are not required to restart Application when you edit the ui_swift_message_types_customer.properties file.
X1 (Institution BIC11)	The name of the sending institution. Optional. **Note:** Only displayed for SAA XML Format 2.
X2 (Department or Application Name)	The name of the sending department or application name. Optional. **Note:** Only displayed for SAA XML Format 2.
X3 (Routing Information/Last Name)	The routing information or last name of the sender. Optional. **Note:** Only displayed for SAA XML Format 2.
X4 (First Name)	The first name of the sender. Optional. **Note:** Only displayed for SAA XML Format 2.
Financial Institution	The financial institution for the sender. Optional. **Note:** Only displayed for SAA XML Format 2.
Branch Information	The branch information for the sender. Optional. **Note:** Only displayed for SAA XML Format 2.
City Name	The name of the city for the sender. Optional. **Note:** Only displayed for SAA XML Format 2.

Field or Check Box	Description
Location	The location code for the sender. Optional. **Note:** Only displayed for SAA XML Format 2.
Country Code	The country code for the sender. Optional. **Note:** Only displayed for SAA XML Format 2.
X1 (Institution BIC11)	The name of the receiving institution. Optional. **Note:** Only displayed for SAA XML Format 2.
X2 (Department or Application Name)	The name of the receiving department or application name. Optional. **Note:** Only displayed for SAA XML Format 2.
X3 (Routing Information/Last Name)	The routing information or last name of the receiver. Optional. **Note:** Only displayed for SAA XML Format 2.
X4 (First Name)	The first name of the receiver. Optional. **Note:** Only displayed for SAA XML Format 2.
Financial Institution	The financial institution for the receiver. Optional. **Note:** Only displayed for SAA XML Format 2.
Branch Information	The branch information for the receiver. Optional. **Note:** Only displayed for SAA XML Format 2.
City Name	The name of the city for the receiver. Optional. **Note:** Only displayed for SAA XML Format 2.
Location	The location code for the receiver. Optional. **Note:** Only displayed for SAA XML Format 2.
Country Code	The country code for the receiver. Optional. **Note:** Only displayed for SAA XML Format 2.
Routing Code	The routing code. Optional. **Note:** Only displayed for SAA XML Format 2.
Validation level	The level of validation. Required. Valid values are None (use SAA default), Minimum, Intermediate, Maximum. **Note:** Only displayed for SAA XML Format 2.
Allow modification in SAA	Whether to allow modification in SAA. Required. Valid values are SAA Default (this is the default), Yes, and No. **Note:** Only displayed for SAA XML Format 2.
Include routing instructions	Whether to include routing instructions. Required. Valid values are Yes and No (default). **Note:** Only displayed for SAA XML Format 2.
Routing Function	The routing function. Required. Valid values are Route (default), DisposeToRoutingPoint, and DisposeToRoutingStep. **Note:** Only displayed for SAA XML Format 2, if Include routing instructions is set to Yes.

Field or Check Box	Description
Routing Point	The routing point. Optional. **Note:** Only displayed for SAA XML Format 2, if Include routing instructions is set to Yes.
Routing Step	The routing step. Optional. Valid values are None (default), Verify, Authorize, Modify, and ReadyToSend. **Note:** Only displayed for SAA XML Format 2, if Include routing instructions is set to Yes.
Network Priority	The network priority. Optional. Valid values are None (default), Normal, System, and Urgent. **Note:** Only displayed for SAA XML Format 2.
Is Possible Duplicate	Whether the trailer is a possible duplicate. Required. Valid values are Yes and No (default). **Note:** Only displayed for SAA XML Format 2.
Request Notification	Whether delivery notification is requested. Required. Valid values are Yes and No (default). **Note:** Only displayed for SAA XML Format 2. **Note:** Application does not mark a message as accepted until the delivery notification or report has been received. Therefore, the Delivery Notification information is stored until the Transmission Report associating the SenderReference with the appropriate Reconciliation information is received.
Service	The network service used. Optional. **Note:** Only displayed for SAA XML Format 2.
User Priority (FIN only; header field 113)	The user priority. Optional. **Note:** Only displayed for SAA XML Format 2.
Value Added Service (FIN only)	The value-added network service used. Optional. **Note:** Only displayed for SAA XML Format 2.
Validation Identifier (FIN only; header field 119)	The validation identifier. Optional. **Note:** Only displayed for SAA XML Format 2.
IsSigningRequested flag (ignored for FIN)	Whether or not a signature is requested as a security option. Optional. Valid values are None (use SAA emission profile configuration—this is the default), True, and False. **Note:** Only displayed for SAA XML Format 2.
Request non-repudiation (SWIFTNet only)	Whether or not non-repudiation is requested as a security option. Optional. Valid values are None (use SAA emission profile configuration—this is the default), True, and False. **Note:** Only displayed for SAA XML Format 2.
PAC Value	??? Optional **Note:** Only displayed for SAA XML Format 2.

Field or Check Box	Description
Sign the DataPDU	Whether to sign the protocol data unit (PDU) for a signature. Optional. Valid values are Yes or No (default). **Note:** Only displayed if Envelope Format is set to either wildcard or SAA XML Format 2 is selected. The signature is optional.
Key Part 1	First part of the validation signature.
Key Part 2	Second part of the validation signature.
Service Identifier	A two-character numeric field indicating the type of data. The default is 01 for all Application and user-to-user messages, 21 for acknowledgements, and 03 for SELECT commands. Required.
Message Priority	Specify the priority of the message delivery. Optional. Valid values are: ◆ Normal (default) ◆ Urgent ◆ System
Delivery Monitoring	One digit that indicates how monitoring will be performed by the SWIFT network. Optional. Valid values are: ◆ No Delivery Monitoring (default) ◆ 1 (Warning Message) indicates that a warning message will be given if the message is not delivered within a reasonable period ◆ 2 (Delivery Notification) indicates notification when the message is delivered ◆ 3 (Non-Delivery Warning and Delivery Notification) indicates that both the non-delivery warning and delivery notification will be given **Note:** The allowable options are tied to the **Message Priority**: priority **Urgent** requires that the user select 1 or 3, priority **Normal** enables the user to select 2 or No Delivery Monitoring.
Obsolescence Period	Indicates the time after which a Delayed Message trailer will be added to the message by the SWIFT network if it has not yet been delivered. This is also the period after which a non-delivery warning will be generated by the SWIFT network, if the appropriate choice for the **Delivery Monitoring** parameter. Optional. **Note:** This parameter must be three numerics. Each unit represents five minutes (so, for example, 003 equates to 15 minutes). SWIFT requires leading zeros, so the if the number of minutes is less than 3 digits, you must include leading zeroes.
FIN Copy Service Code (tag 103):	A typical configuration requires that the FIN Copy Service Code tag be included in the envelope (usually set to **COP**). Optional. The SWIFT network support the FIN Copy mode, in which a message is sent to an intermediary for approval before it goes to its final destination (or is just copied to the intermediary without requiring approval).
Banking Priority (tag 113):	A four-character optional tag indicating the banking priority. The allowed values are agreed on by you and your trading partner or partners. Optional.

Field or Check Box	Description
Validation Flag (tag 119):	Specifies the validation flag. Optional. This is an optional part of the header that can contain a code word to indicate that certain types of validations should be performed on the enveloped message. The valid values for this tag vary depending on the message type.
Payment Release Information (tag 115):	Specifies the payment release information. This is an envelope component used in FIN Copy that contains information from the central institution to the receiver of the payment message. The information from this parameter will be placed by the SWIFT network into the MT 097 FIN Copy Message Authorization/Refusal Notification in Y-copy mode.
Include Possible Duplicate Emission (PDE) Trailer	Indicates whether to include a trailer specifying that this message may be a duplicate. Required. This is an optional component of the envelope that indicates that this message may already have been sent. For example, if the system crashes during the delivery of a message, another copy of the message could be sent, with this trailer included to indicate that it may be a duplicate. Valid values are Yes and No (default).
Include Training (TNG) Trailer	Specifies whether to include a training (TNG) trailer. Required. This is an optional component of the envelope that indicates the message contained is being sent for system testing purposes. Valid values are Yes and No (default).
Sender Logical Terminal	Specifies the logical terminal for outbound FIN. This is a single alphanumeric character Required. **Note:** This parameter is necessary to send messages through the WebSphere MQ adapter using the FileAct protocol.
* Sender Branch Code	Three-character code that further refines the Sender ID. For example, if the SenderID is ROMANSFL, the branch code might be WAS to indicate the Washington branch. This parameter acts as the Sender ID qualifier for envelope matching. Required.
* Receiver Branch Code	Three-character code that further refines the Receiver ID. For example, if the ReceiverID is ROMANSFL, the branch code might be WAS to indicate the Washington branch. This parameter acts as the Receiver ID qualifier for envelope matching. Required.
Maximum Message Size	The maximum size of message that is allowed. The default is 10,000. This parameter only accepts numeric input. Only displayed if you selected Yes for the **Enforce Message Size Limit** parameter. Required.

Field or Check Box	Description
Use global control number	Whether to use a global control number. Required. Valid values are: ♦ Yes (default) ♦ Yes (and generate from data) **Note:** If you select this option, you must select at least one parameter for Primary Name Format. ♦ No Displayed only if **Message User Reference** is set to Use Control Number.
Primary Name Format	Check boxes to instruct what information to include when generating a name for a primary global control number and finding the correct number to assign based on that name. Optional. Select from: ♦ Sender ID ♦ Sender Branch Code ♦ Receiver ID ♦ Receiver Branch Code ♦ Message Format (MT/MX) ♦ Message Type ♦ Validation Flag (MT only) ♦ Envelope Format (FIN/XML2) **Note:** If you are using a generated global control number (**Use global control number** is set to Yes (and generate name from data)), Application tries to generate and match the following control numbers: ♦ First, it tries to generate and match the control number in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the control number SWIFT_100_Outbound_FIN. ♦ Second, it tries to generate and match the First Backup Name. ♦ Third, it tries to generate and match the Second Backup Name. ♦ Fourth, if there is not an existing control number that matches the names generated in steps 1-3, a control number with the name assigned in the first step is created.

Field or Check Box	Description
First Backup Name Format	The first backup name format to use when generating a global control number. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional.
	Select from:
	◆ Sender ID
	◆ Sender Branch Code
	◆ Receiver ID
	◆ Receiver Branch Code
	◆ Message Format (MT/MX)
	◆ Message Type
	◆ Validation Flag (MT only)
	◆ Envelope Format (FIN/XML2)
	Note: If you are using a generated global control number (**Use global control number** is set to Yes (and generate name from data)), Application tries to generate and match the following control numbers:
	◆ First, it tries to generate and match the control number in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the control number SWIFT_100_Outbound_FIN.
	◆ Second, it tries to generate and match the First Backup Name.
	◆ Third, it tries to generate and match the Second Backup Name.
	◆ Fourth, if there is not an existing control number that matches the names generated in steps 1-3, a control number with the name assigned in the first step is created.

Field or Check Box	Description
Second Backup Name Format	The second backup name format to use when generating a global control number. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional.
	Select from:
	◆ Sender ID
	◆ Sender Branch Code
	◆ Receiver ID
	◆ Receiver Branch Code
	◆ Message Format (MT/MX)
	◆ Message Type
	◆ Validation Flag (MT only)
	◆ Envelope Format (FIN/XML2)
	Note: If you are using a generated global control number (**Use global control number** is set to Yes (and generate name from data)), Application tries to generate and match the following control numbers:
	◆ First, it tries to generate and match the control number in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the control number SWIFT_100_Outbound_FIN.
	◆ Second, it tries to generate and match the First Backup Name.
	◆ Third, it tries to generate and match the Second Backup Name.
	◆ Fourth, if there is not an existing control number that matches the names generated in steps 1-3, a control number with the name assigned in the first step is created.
Global Message User Reference (MUR) Control Number	Select the control number with which to start generating control numbers. Optional.
	Displayed only if **Message User Reference** is set to Use Control Number.
Process Data element from which to set the Message User Reference	Specify the process data element from which the MUR will be set.
	Displayed only if **Message User Reference** is set to Use value from process data. Required.
Local Message User Reference (MUR)	The message user reference block is an optional section in the SWIFT envelope that the SWIFT network tracks for acknowledgement purposes. If you choose to use a local control number as the user reference, type the starting value of that control number in this parameter.
Expect an acknowledgement for messages sent using this envelope	Whether to expect an acknowledgement for messages that are sent using this envelope. Valid values are Yes and No (default). Required.
Acknowledgement overdue after (hours)	Amount of time, in hours, within which you must receive an acknowledgement. Valid value is four numeric characters. Optional.

Field or Check Box	Description
Acknowledgement overdue after (minutes)	Amount of time, in minutes, within which you must receive an acknowledgement. Valid value is four numeric characters. Optional.
Translate documents prior to enveloping	Whether to translate the documents prior to enveloping them. Valid values are Yes or No (default). Required.
Map Name Mode	How to determine which map to use to translate the message. Required. Valid values are: ◆ Specify (default) ◆ Generate from data (this is useful if you want to share across envelopes) **Note:** If you select this option, you must select at least one parameter for Primary Name Format. This parameter is displayed only if **Translate document prior to enveloping** set to Yes.
Map Name	Which map to use to perform a compliance check (if **Translate document prior to enveloping** set to Yes and **Map Name Mode** set to Yes). The map must already be checked in to Application. Optional/
Primary Name Format	Check boxes to instruct what information to include when generating and matching a name for the map. Optional. Select from: ◆ Sender ID ◆ Sender Branch Code ◆ Receiver ID ◆ Receiver Branch Code ◆ Message Format (MT/MX) ◆ Message Type ◆ Validation Flag (MT only) ◆ Envelope Format (FIN/XML2) **Note:** If you are using a generated map name (**Map Name Mode** is set to Generate from data), Application tries to generate and match the following maps: ◆ First, it tries to generate and match the map in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the control number SWIFT_100_Outbound_FIN. ◆ Second, it tries to generate and match the First Backup Name. ◆ Third, it tries to generate and match the Second Backup Name. ◆ Fourth, if there is not an existing map that matches the names generated in steps 1-3, an error is generated. **Note:** Only occurs if **Generate an error if no matching map is found** is set to Yes.

Field or Check Box	Description
First Backup Name Format	The first backup name format to use when generating a map name. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional.
	Select from:
	◆ Sender ID
	◆ Sender Branch Code
	◆ Receiver ID
	◆ Receiver Branch Code
	◆ Message Format (MT/MX)
	◆ Message Type
	◆ Validation Flag (MT only)
	◆ Envelope Format (FIN/XML2)
	Note: If you are using a generated map name (**Map Name Mode** is set to Generate from data), Application tries to generate and match the following maps:
	◆ First, it tries to generate and match the map in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the control number SWIFT_100_Outbound_FIN.
	◆ Second, it tries to generate and match the First Backup Name.
	◆ Third, it tries to generate and match the Second Backup Name.
	◆ Fourth, if there is not an existing map that matches the names generated in steps 1-3, an error is generated.
	Note: Only occurs if **Generate an error if no matching map is found** is set to Yes.

Field or Check Box	Description
Second Backup Name Format	The second backup name format to use when generating a map name. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional.
	Select from:
	◆ Sender ID
	◆ Sender Branch Code
	◆ Receiver ID
	◆ Receiver Branch Code
	◆ Message Format (MT/MX)
	◆ Message Type
	◆ Validation Flag (MT only)
	◆ Envelope Format (FIN/XML2)
	Note: If you are using a generated map name (**Map Name Mode** is set to Generate from data), Application tries to generate and match the following maps:
	◆ First, it tries to generate and match the map in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the control number SWIFT_100_Outbound_FIN.
	◆ Second, it tries to generate and match the First Backup Name.
	◆ Third, it tries to generate and match the Second Backup Name.
	◆ Fourth, if there is not an existing map that matches the names generated in steps 1-3, an error is generated.
	Note: Only occurs if **Generate an error if no matching map is found** is set to Yes.
Generate an error if no matching map is found	Specifies whether to generate an error if the Map Name selected is not found. Valid values are Yes (default) and No.
	Note: Only displayed if **Map Name Mode** is set to Generate from data.

Field or Check Box	Description
On a translation error, determine the Business Process by	How to determine the business process name to use if there were errors in the translation process. Required. Valid values are: ◆ Specifying a Business Process name (default) ◆ Generating a Business Process name from the data (this is useful if you want to share across envelopes) **Note:** If you select this option, you must select at least one parameter for Primary Name Format. **Note:** If you select a translation error business process, when a document with compliance errors is encountered an instance of the error business process is generated, using the non-compliant document as the primary document. The processing of that document within the enveloper is halted at that point, and the next document starts processing. If you do not specify a translation error business process, the enveloper continues to process the non-compliant document. In either case, the status report containing the translation errors is created. If HALT_ON_TRANS_ERROR is set to Yes or True in ProcessData, no more documents are processed after a non-compliant document is encountered. If HALT_ON_TRANS_ERROR is set to No or False in ProcessData, the rest of the documents will be processed after a non-compliant document is encountered. If HALT_ON_TRANS_ERROR is not defined, the behavior depends on the enveloping mode; IMMEDIATE mode behaves as if HALT_ON_TRANS_ERROR is set to True, and DEFERRED mode behaves as if HALT_ON_TRANS_ERROR is set to False.

Field or Check Box	Description
Primary Name Format	Check boxes to instruct what information to include when generating and matching a name for the error business process name. Optional. Select from: ◆ Sender ID ◆ Sender Branch Code ◆ Receiver ID ◆ Receiver Branch Code ◆ Message Format (MT/MX) ◆ Message Type ◆ Validation Flag (MT only) ◆ Envelope Format (FIN/XML2) **Note:** If you are using a generated error business process name (**On a translation error, determine Error Business Process Name By** is set to Generating a Business Process name from the data), Application tries to generate and match the following business process: ◆ First, it tries to generate and match the business process in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the business process SWIFT_100_Outbound_FIN. ◆ Second, it tries to generate and match the First Backup Name. ◆ Third, it tries to generate and match the Second Backup Name. ◆ Fourth, if there is not an existing business process that matches the names generated in steps 1-3, an error is generated. **Note:** Only occurs if **Generate an error if no generated business process name exists in the system** is set to Yes.

Field or Check Box	Description
First Backup Name Format	The first backup name format to use when generating an error business process name. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional.

Select from:

- Sender ID

- Sender Branch Code

- Receiver ID

- Receiver Branch Code

- Message Format (MT/MX)

- Message Type

- Validation Flag (MT only)

- Envelope Format (FIN/XML2)

Note: If you are using a generated error business process name (**On a translation error, determine Error Business Process Name By** is set to Generating a Business Process name from the data), Application tries to generate and match the following business process:

- First, it tries to generate and match the business process in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the business process SWIFT_100_Outbound_FIN.

- Second, it tries to generate and match the First Backup Name.

- Third, it tries to generate and match the Second Backup Name.

- Fourth, if there is not an existing business process that matches the names generated in steps 1-3, an error is generated.

Note: Only occurs if **Generate an error if no generated business process name exists in the system** is set to Yes.

Field or Check Box	Description
Second Backup Name Format	The second backup name format to use when generating an error business process name. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional.
	Select from:
	◆ Sender ID
	◆ Sender Branch Code
	◆ Receiver ID
	◆ Receiver Branch Code
	◆ Message Format (MT/MX)
	◆ Message Type
	◆ Validation Flag (MT only)
	◆ Envelope Format (FIN/XML2)
	Note: If you are using a generated error business process name (**On a translation error, determine Error Business Process Name By** is set to Generating a Business Process name from the data), Application tries to generate and match the following business process:
	◆ First, it tries to generate and match the business process in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the business process SWIFT_100_Outbound_FIN.
	◆ Second, it tries to generate and match the First Backup Name.
	◆ Third, it tries to generate and match the Second Backup Name.
	◆ Fourth, if there is not an existing business process that matches the names generated in steps 1-3, an error is generated.
	Note: Only occurs if **Generate an error if no generated business process name exists in the system** is set to Yes.
Generate an error if no generated business process name exists in the system	Specify whether to generate an error if there is no match to the generated business process name in the system. Valid values are Yes (default) and No.
	Note: Only displayed if **On a translation error, determine the Error Business Process By** is set to Generating the business process name from the data.
Business Process List	Select a previously created business process to associate with this envelope.
	Displayed only if **On a translation error, determine the Business Process by** is set to Specifying a business process names. Optional.

Field or Check Box	Description
Determine the Business Process By	How to determine the business process name to use if there were no errors in the compliance check. Required. Valid values are: ◆ Specify a Business Process (default) ◆ Generate Business Process Name (from the data) **Note:** If you select this option, you must select at least one parameter for Primary Name Format.
Primary Name Format	Check boxes to instruct what information to include when generating and matching a name for the business process. Optional. Select from: ◆ Sender ID ◆ Sender Branch Code ◆ Receiver ID ◆ Receiver Branch Code ◆ Message Format (MT/MX) ◆ Message Type ◆ Validation Flag (MT only) ◆ Envelope Format (FIN/XML2) **Note:** If you are using a generated business process name (**Determine the Business Process By** is set to Generating the business process name from the data), Application tries to generate and match the following business process: ◆ First, it tries to generate and match the business process in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the business process SWIFT_100_Outbound_FIN. ◆ Second, it tries to generate and match the First Backup Name. ◆ Third, it tries to generate and match the Second Backup Name. ◆ Fourth, if there is not an existing business process that matches the names generated in steps 1-3, an error is generated. **Note:** Only occurs if **Generate an error if no generated business process name exists in the system** is set to Yes.

Field or Check Box	Description
First Backup Name Format	The first backup name format to use when generating a business process name. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional.
	Select from:
	◆ Sender ID
	◆ Sender Branch Code
	◆ Receiver ID
	◆ Receiver Branch Code
	◆ Message Format (MT/MX)
	◆ Message Type
	◆ Validation Flag (MT only)
	◆ Envelope Format (FIN/XML2)
	Note: If you are using a generated business process name (**Determine the Business Process By** is set to Generating a Business Process name from the data), Application tries to generate and match the following business process:
	◆ First, it tries to generate and match the business process in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the business process SWIFT_100_Outbound_FIN.
	◆ Second, it tries to generate and match the First Backup Name.
	◆ Third, it tries to generate and match the Second Backup Name.
	◆ Fourth, if there is not an existing business process that matches the names generated in steps 1-3, an error is generated.
	Note: Only occurs if **Generate an error if no generated business process name exists in the system** is set to Yes.

Field or Check Box	Description
Second Backup Name Format	The second backup name format to use when generating a business process name. The system allows for two alternatives if the Primary Name Format is not found—checking for the First Backup Name Format and then, if that is not found, checking for the Second Backup Name Format. Optional.
	Select from:
	◆ Sender ID
	◆ Sender Branch Code
	◆ Receiver ID
	◆ Receiver Branch Code
	◆ Message Format (MT/MX)
	◆ Message Type
	◆ Validation Flag (MT only)
	◆ Envelope Format (FIN/XML2)
	Note: If you are using a generated business process name (**Determine the Business Process By** is set to Generating a Business Process name from the data), Application tries to generate and match the following business process:
	◆ First, it tries to generate and match the business process in the primary name format (replacing the values that are selected for the corresponding value in the message). For example, if only message type is selected, and the message type is 100, Application checks for the business process SWIFT_100_Outbound_FIN.
	◆ Second, it tries to generate and match the First Backup Name.
	◆ Third, it tries to generate and match the Second Backup Name.
	◆ Fourth, if there is not an existing business process that matches the names generated in steps 1-3, an error is generated.
	Note: Only occurs if **Generate an error if no generated business process name exists in the system** is set to Yes.
Generate an error if no generated business process name exists in the system	Specify whether to generate an error if there is no match to the generated business process name in the system. Required. Valid values are Yes (default) and No.
	Note: Only displayed if **Determine the Business Process By** is set to Generate Business Process Name.
Business Process List	Select a previously created business process to associate with this envelope. Optional.
	Displayed only if **Determine the Business Process Name** is set to Specify Business Process.

Field or Check Box	Description
Extraction Options	Business process data extraction. Required. Valid values: ◆ Determined by business process (default) ◆ Extract to a file system directory ◆ Extract to a mailbox
Data Extraction Directory	Directory for data extraction. Displayed only if **Extraction Options** set to Extract to a file system directory. Optional.
Data Extraction Filename	Filename for data extraction. Displayed only if **Extraction Options** set to Extract to a file system directory. Optional.
Data Extraction Mailbox	Mailbox for data extraction. Displayed only if **Extraction Options** set to Extract to a mailbox. Optional.
Data Extraction Mailbox Message Name	Mailbox message name for data extraction. Displayed only if **Extraction Options** set to Extract to a mailbox. Optional.

Configuring the EDI Encoder Service for SWIFTNet Outbound Messages

The EDI Encoder service determines which envelope will be used on the document. If translations are specified in an envelope, the service determines which map to use. For SWIFTNet outbound messages you need to configure five parameters to allow the outbound message to be sent correctly. See *EDI Encoder Service* for more information.

To configure the EDI Encoder service, you must specify settings for the following fields in the GPM that match the values you configured in the outbound envelope:

✦ AccepterLookupAlias

✦ ReceiverID

✦ SenderID

✦ ReceiverIDQual

✦ SenderIDQual

You need to set the sender and receiver variables based on the desired source and destination for the outbound message. These variables also match the corresponding values in the envelope, except that the envelope allows wildcards (which will match documents encoded with any value for that variable), whereas the value used in the EDI encoder must always be the full eight-character identifier and three-character branch code of the desired source and destination.

Or, if you are editing the EDI Encoder BPML, you need to include these parameters and values, where for xxx you substitute the values set up in the outbound envelope:

```
<operation name="EDI Encode">
  <participant name="EDIEncoder"/>
  <output message="EDIEnc_In">
    <assign to="AccepterLookupAlias">xxx</assign>
```

```
        <assign to="ReceiverID">xxx</assign>
        <assign to="SenderID">xxx</assign>
        <assign to="ReceiverIDQual">xxx</assign>
        <assign to="SenderIDQual">xxx</assign>
        <assign to="." from="*"/>
    </output>
    <input message="EDIEnc_Out">
        <assign to="." from="*"/>
    </input>
</operation>
```

SWIFT Business Processes

Overview

To help you accomplish your business goals, Application provides two predefined enveloping business processes, which are used by Application to implement SWIFT processing: SWIFTEnvelope and SWIFTDeenvelope. These predefined business processes are initiated by other processes/services during SWIFT processing and do not require you to modify them.

Application also provides predefined business processes, which are used by Application to implement SWIFT transport, including the SWIFTNetClient business process (using for InterAct transport), and the SWIFTNet ClientFA (used for FileAct transport) business process. These predefined business processes are initiated by other processes/services during SWIFT processing but do require you to modify them.

The following table lists business goals for some of the predefined SWIFT business processes:

Business Process	Business Goals
SWIFTDeenvelope	Extracts SWIFT message types from a message and translates and processes them, according to the content of the envelopes.
SWIFTEnvelope	Applies a SWIFT envelope to one or more SWIFT message types and then uses the envelope data to translate and process them.
SWIFTNetClient	Used for InterAct processing. Contains the necessary parameters so the SWIFTNet Client service can prepare the request and send it (outbound) to the Application SWIFTNet MEFG Server. The client application on the SWIFTNet MEFG Server processes this request, performs the necessary communication exchange with the SWIFTNet SAG/SNL instance, and sends the request to the SWIFTNet Network.
SWIFTNetClientFA	Used for FileAct processing. Contains the necessary parameters so the SWIFTNet Client service can prepare the request and send it (outbound) to the Application SWIFTNet MEFG Server. The client application on the SWIFTNet MEFG Server processes this request, performs the necessary communication exchange with the SWIFTNet SAG/SNL instance, and sends the request to the SWIFTNet Network.
handleSWIFTNetServerRequest	Used for InterAct processing. Enables Application to receive SWIFTNet messages. This is the bootstrap business process used by the inbound SWIFTNet request through the Application SWIFTNet MEFG Server. It is a system business process used by the SWIFTNet Server Adapter, which pre-processes the incoming request, search the SWIFTNet Routing Rule table, and route the request payload to the business process for processing.

Business Process	Business Goals
handleSWIFTNetServerSnFRequest	Used for InterAct processing. Enables Application to receive SWIFTNet store-and-forward messages. This is a bootstrap business process used by the inbound SWIFTNet request that includes a store-and-forward option. For the store-and-forward option, an incoming request is not processed immediately, but instead is stored in the responder mailbox in Application to be responded to later. Application then sends an acknowledgement that the request has been successfully stored to the requestor through the Application SWIFTNet MEFG Server.
handleSWIFTNetServerFARequest	Used for FileAct processing. Enables Application to receive SWIFTNet messages. This is the bootstrap business process used by the inbound SWIFTNet request through the Application SWIFTNet MEFG Server. It is a system business process used by the SWIFTNet Server Adapter, which pre-processes the incoming request, searches the SWIFTNet Routing Rule table, and routes the request payload to the business process for processing.
handleSWIFTNetServerFASnF Request	Used for FileAct processing. Enables Application to receive SWIFTNet store-and-forward messages. This is a bootstrap business process used by the inbound SWIFTNet request that includes a store-and-forward option. For the store-and-forward option, an incoming request is not processed immediately, but instead is stored in the responder mailbox in Application to be responded to later. Application then sends an acknowledgement that the request has been successfully stored to the requestor through the Application SWIFTNet MEFG Server.
handleSWIFTNetServerFAEvent	The handleSWIFTNetServerFAEvent business process is used by the SWIFTNet MEFG Server. It is a system business process called by the SWIFTNet Server adapter that preprocesses the incoming FileAct Event with a COMPLETED status, searches the SWIFTNet routing rule table and bootstraps the business process for processing
SWIFTMessageEntryEnvelope	This is the default business process used to look up the correct envelope, based on information provided by the user while configuring the Send information in the SWIFT Message Entry Workstation. This business process uses the EDI Encoder service, Envelope service, and then invokes the business process specified in the SWIFT Outbound Envelope.

SWIFTDeenvelope Business Process

The SWIFTDeenvelope business process is used to deenvelope SWIFT data. A typical scenario is one in which SWIFT data must be received from a trading partner. The data must be deenveloped to extract identifying batch and interchange data, and the SWIFTDeenvelope business process helps to provide the deenveloping services.

The SWIFTDeenvelope business process is initiated as part of the following inbound process flow:

1. You create a business process that calls the EDI Deenveloping service.

2. The EDI Deenveloping service parses the whole document and extracts messages from it so it can be further processed by Application.

3. Passes the SWIFT messages to the SWIFTDeenvelope business process. The Generic Deenvelope service runs as a subprocess.

4. The SWIFTDeenvelope business process looks up the envelope, based on the data found in the header.

5. The envelope specifies what to do with the deenveloped messages.

6. Starts the Invoke Business Process service or a subprocess service to initiate the appropriate business process to handle each deenveloped message.

If exceptions occur when running an EDI Deenveloping business process, Application generates an EDI Compliance Report.

This table lists the configuration parameters for the SWIFTDeenvelope business process:

Parameter	Default	Description
Document Tracking	False	When document tracking is enabled for a business process, tracking information is carried with the message throughout the process, and the tracking information is persisted about the message regardless of the persistence level you configured globally for Application.
Set onfault processing	False	Onfault processing allows the process to immediately execute the on-fault activity specified in the process, even if the process has not yet reached that step in the process. For example, if a process fails at step 3, but the on-fault activity is specified in step 7, if onfault processing is enabled, the process proceeds to the step 7 on-fault rather than halting at step 3.
Queue	4	Application enables you to set performance optimizations by queue, defining queue levels to allocate resources. This number indicates the previously allocated queue level that you want for this business process model for processing.
Persistence Level	System Default	The level of data to retain for generating a status report that describes each step that the business process completes. System default indicates that, for the data, configuration is already defined in Application to retain data.
Recovery Level	Manual	The level of recovery for this business process if the business process should halt during execution. Manual requires you to resume or restart the business process manually.
Document Storage Type	System Default	The level of document storage for messages that process when the business process runs. System Default specifies to store messages in the file system or database, according to how you configured archiving and purging in Application.
Life Span	Life Span Days — 2 Life Span Hours — 0 Life Span Type — System Level Removal Method — Archive	The length of time, in days and hours, to retain the data in Application, along with the life span type and removal method.

Parameter	Default	Description
Complete by Deadline	None Available **Note:** To set a deadline you must change it in the business process.	Complete by – The deadline time, in hours and minutes, by which the business process must complete process once it starts. ◆ First Notification: Hours and Minutes – Whether to receive notification before a business process deadline. ◆ Second Notification: Hours and Minutes – Whether to receive another notification before a business process deadline.
Event Reporting Level	Full	The level of event reporting that is retrieved for this business process when it runs. Full specifies to generate events for the business process, including the business process start and end time, start and end times for all services or services running as a result of this business processes, and any resulting errors and exceptions.

The following BPML code makes up the SWIFTDeenvelope business process:

```
<process name="SWIFTDeenvelope">
  <rule name="contract">
    <condition>CONTRACT_FOUND = &quot;YES&quot;</condition>
  </rule>
  <sequence>
    <assign to="RunInValidationMode">FALSE</assign>
   <sequence>
   <operation>
     <participant name="DeenvelopeGeneric"/>
     <output message="Xout">
   <assign to="break_processor">SWIFT</assign>
   <assign to="." from="*"></assign>
     </output>
     <input message="Xin">
   <assign to="." from="*"></assign>
     </input>
   </operation>
       <onFault code="Interchange-Non-Compliant">
         <!-- Just catch the error, so that the BP will continue. -->
         <!-- Dummy assign, since we need something here. -->
         <assign to="BREAK">YES</assign>
       </onFault>
       <onFault code="Transaction-Non-Compliant">
         <!-- Just catch the error, so that the BP will continue. -->
         <!-- Dummy assign, since we need something here. -->
         <assign to="BREAK">YES</assign>
       </onFault>
       <onFault>
         <operation>
           <participant name="BPExceptionService"/>
           <output message="Xout" >
             <assign to="exceptionCode" from="Prev_NotSuccess_Adv_Status/text()"/>
           </output>
           <input message="Xin"/>
```

```
            </operation>
          </onFault>
      </sequence>
       <choice>
         <select>
           <case ref="contract" activity="invoke_contract_workflow"/>
         </select>
         <sequence name="invoke_contract_workflow">
           <operation>
             <participant name="InvokeBusinessProcessService"/>
             <output message="Xout">
               <assign to="INVOKE_MODE">ASYNC</assign>
               <assign to="." from="*"></assign>
             </output>
             <input message="Xin">
               <assign to="." from="*"></assign>
             </input>
           </operation>
         </sequence>
       </choice>
    </sequence>
</process>
```

Before Using the SWIFTDeenvelope Business Process

Before you use the SWIFTDeenvelope business process, you must complete the following task:

1. Create a SWIFT inbound envelope for each SWIFT message type that you are receiving. See *Inbound SWIFT envelope* on page 19.

SWIFTEnvelope Business Process

The SWIFTEnvelope business process is initiated when it is called by another business process. The SWIFTEnvelope business process envelopes the SWIFT messages contained in the business process context with outbound SWIFT envelopes that you have preconfigured. You must have created one outbound SWIFT envelope for each SWIFT message type that you are sending.

A typical scenario is one in which SWIFT data must be sent to a trading partner. To prepare for this, the data must be enveloped to provide identifying batch and interchange data. The SWIFTEnvelope business process helps to provide these enveloping services.

The SWIFTEnvelope business process is initiated as part of the following outbound process flow:

1. You create a business process that calls the EDI Encoder service or Document Extraction service.

2. The EDI Encoder service or Document Extraction service looks up the envelope to apply and prepares the document to be enveloped.

3. Either the business process calls the EDI Enveloping service or the Document Extraction service is configured to perform enveloping.

4. The EDI Enveloping service or Document Extraction service starts the SWIFTEnvelope business process (which runs the Generic Envelope service as a subprocess to extract the name of the business process).

5. The SWIFTEnvelope business process searches the envelope definition to retrieve information to envelope each message.

Note: The Sender ID, Receiver ID, and Lookup Alias in your Outbound Envelope definition must match the parameters that you define for this outbound business process.

This table lists the configuration parameters for the SWIFTEnvelope business process:

Parameter	Default	Description
Document Tracking	False	When document tracking is enabled for a business process, tracking information is carried with the message throughout the process, and the tracking information is persisted about the message regardless of the persistence level you configured globally for Application.
Set onfault processing	False	Onfault processing allows the process to immediately execute the on-fault activity specified in the process, even if the process has not yet reached that step in the process. For example, if a process fails at step 3, but the on-fault activity is specified in step 7, if onfault processing is enabled, the process proceeds to the step 7 on-fault rather than halting at step 3.
Queue	4	Application enables you to set performance optimizations by queue, defining queue levels to allocate resources. This number indicates that previously allocated queue level that you want for this business process model for processing.
Persistence Level	System Default	The level of data to retain for generating a status report that describes each step that the business process completes. System default indicates that, for the data, configuration is already defined in Application to retain data.
Recovery Level	Manual	The level of recovery for this business process if the business process should halt during execution. Manual requires you to resume or restart the business process manually.
Document Storage Type	System Default	The level of document storage for messages that process when the business process runs. System Default specifies to store messages in the file system or database, according to how you configured archiving and purging in Application.
Life Span	Life Span Days — 2 Life Span Hours — 0 Life Span Type — System Level Removal Method — Archive	The length of time, in days and hours, to retain the data in Application, along with the life span type and removal method.

Parameter	Default	Description
Complete by Deadline	None Available **Note:** To set a deadline you must change it in the business process.	Complete by – The deadline time, in hours and minutes, by which the business process must complete process once it starts. ◆ First Notification: Hours and Minutes – Whether to receive notification before a business process deadline. ◆ Second Notification: Hours and Minutes – Whether to receive another notification before a business process deadline.
Event Reporting Level	Full	The level of event reporting that is retrieved for this business process when it runs. Full specifies to generate events for the business process, including the business process start and end time, start and end times for all services or services running as a result of this business processes, and any resulting errors and exceptions.

The following BPML code makes up the SWIFT Envelope business process:

```
<process name="SWIFTEnvelope">

  <rule name="contract">
    <condition>CONTRACT_FOUND = &quot;YES&quot;</condition>
  </rule>

  <sequence>
    <operation>
      <participant name="EnvelopeGeneric" />
      <output message="Xout" >
        <assign to="." from="*"></assign>
      </output>
      <input message="Xin" >
        <assign to="." from="*"></assign>
      </input>
    </operation>

    <choice>
      <select>
        <case ref="contract" activity="invoke_contract_workflow" />
      </select>
      <sequence name="invoke_contract_workflow">
        <operation>
          <participant name="InvokeBusinessProcessService" />
          <output message="Xout">
            <assign to="INVOKE_MODE">ASYNC</assign>
            <assign to="." from="*"></assign>
          </output>
          <input message="Xin" >
            <assign to="." from="*"></assign>
          </input>
        </operation>
      </sequence>
    </choice>
```

```
        </sequence>
</process>
```

Before Using the SWIFTEnvelope Business Process

Before you use the SWIFTEnvelope business process, you must complete the following task:

1. Create a SWIFT outbound envelope for each SWIFT message type that you are sending. See *Outbound SWIFT envelope* on page 36.

SWIFTNetClient Business Process

The SWIFTNet Client service enables you to use Secure Socket Layers (SSL), but to do so you must also upgrade the SWIFTNetClient business process if you have not already done so. The upgrade BPML differs based on whether you are using InterAct or FileAct. See the *SWIFTNet Client Service* documentation for more information on how to upgrade the SWIFTNetClient business process if you are using SSL.

Note: If you previously installed an earlier version of the Application Standards Library, you do not need to upgrade the SWIFTNetClient business process again. However, you will need to reinstall the SWIFTNet MEFG Server (see *SWIFTNet MEFG Server* on page 147 for more information).

The SWIFTNetClient business process contains the necessary parameters so the SWIFTNet Client service can prepare the request and send it to the Application SWIFTNet MEFG Server. The client application on the SWIFTNet MEFG Server processes this request, performs the necessary communication exchange with the SWIFTNet SAG/SNL instance, and sends the request to the SWIFTNet Network.

Note: This business process is used for InterAct processing only. It takes the default parameters configured in the SWIFTNet Client service. If a parameter defined in the SWIFTNet Client Service is specified in this business process, the value in the business process overrides the parameter in the SWIFTNet Client service.

The SWIFTNetClient business process is initiated as part of the following outbound process flow:

1. The SWIFTNetClient business process invokes the SWIFTNet Client service and passes it all the necessary parameters to send a request.

2. The client application on the SWIFTNet MEFG Server processes the request.

This table lists the configuration parameters for the SWIFTNetClient business process:

Parameter	Default	Description
Document Tracking	False	When document tracking is enabled for a business process, tracking information is carried with the message throughout the process, and the tracking information is persisted about the message regardless of the persistence level you configured globally for Application.

Parameter	Default	Description
Set onfault processing	False	Onfault processing allows the process to immediately execute the on-fault activity specified in the process, even if the process has not yet reached that step in the process. For example, if a process fails at step 3, but the on-fault activity is specified in step 7, if onfault processing is enabled, the process proceeds to the step 7 on-fault rather than halting at step 3.
Start mode	async	Asynchronous initiation is selected by default. Starting business processes asynchronously is recommended. Asynchronous mode is standard Application processing, wherein the business process is placed in queue and processed.
Transaction	False	This option instructs Application to treat the entire process as a single transaction so that either all of the steps complete, or, in the event of an error, none of them do. When an error occurs, no data is committed; data returns to its pre-process state. By default, this transaction mode is not enabled.
Queue	4	Application enables you to set performance optimizations by queue, defining queue levels to allocate resources. This number indicates the previously allocated queue level that you want for this business process model for processing.
Persistence Level	System Default	The level of data to retain for generating a status report that describes each step that the business process completes. System default indicates that, for the data, configuration is already defined in Application to retain data.
Recovery Level	Manual	The level of recovery for this business process if the business process should halt during execution. Manual requires you to resume or restart the business process manually.
Document Storage Type	System Default	The level of document storage for messages that process when the business process runs. System Default specifies to store messages in the file system or database, according to how you configured archiving and purging in Application.
Life Span	Life Span Days — 2 Life Span Hours — 0 Life Span Type — System Level Removal Method — Archive	The length of time, in days and hours, to retain the data in Application, along with the life span type and removal method.

Parameter	Default	Description
Complete by Deadline	None Available **Note:** To set a deadline you must change it in the business process.	Complete by – The deadline time, in hours and minutes, by which the business process must complete process once it starts. ◆ First Notification: Hours and Minutes – Whether to receive notification before a business process deadline. ◆ Second Notification: Hours and Minutes – Whether to receive another notification before a business process deadline.
Event Reporting Level	Full	The level of event reporting that is retrieved for this business process when it runs. Full specifies to generate events for the business process, including the business process start and end time, start and end times for all services or services running as a result of this business processes, and any resulting errors and exceptions.

The following BPML code makes up the SWIFTNetClient business process:

Note: The **bold** lines indicate information that you need to modify to match your installation.

```
<process name="SWIFTNetClient">
    <sequence name="SWIFTNetClientService">
        <operation name="set user token">
            <participant name="SetUserToken"/>
            <output message="SetUserTokenMessage">
                <assign to="USER_TOKEN">admin</assign>
                <assign to="." from="*"/>
            </output>
            <input message="inmsg">
                <assign to="." from="*"/>
            </input>
        </operation>
        <!-- build SWIFTNET request -->
        <operation>
            <participant name="SWIFTNetClientService"/>
            <output message="handleClientRequest">
                <assign to="." from="*"/>
            </output>
            <input message="testing">
                <assign to="." from="*"/>
            </input>
        </operation>
    </sequence>
</process>
```

Before Using the SWIFTNetClient Business Process

Before you use the SWIFTNetClient business process, you must complete the following tasks:

1. Configure the SWIFTNet Client service (or create a new instance of it) to reflect your installation. See *SWIFTNet Client Service*.

2. Enable Document Tracking in the Business Process Manager.

    Then you can execute the SWIFTNetClient business process as part of your SWIFTNet processing.

SWIFTNetClientFA Business Process

This is a client business process to send requests to SWIFTNet using FileAct.

Note: This business process is used with FileAct processing only. It takes the default parameters configured in the SWIFTNet Client service. If a parameter defined in the SWIFTNet Client Service is specified in this business process, the value in the business process overrides the parameter in the SWIFTNet Client service.

The SWIFTNetClientFA business process is initiated as part of the following outbound process flow:

1. The SWIFTNetClientFA business process invokes the SWIFTNet Client service and passes it all the necessary parameters to send a request.

2. The client application on the SWIFTNet MEFG Server processes the request.

This table lists the configuration parameters for the SWIFTNetClientFA business process:

Parameter	Default	Description
Document Tracking	False	When document tracking is enabled for a business process, tracking information is carried with the message throughout the process, and the tracking information is persisted about the message regardless of the persistence level you configured globally for Application.
Set onfault processing	False	Onfault processing allows the process to immediately execute the on-fault activity specified in the process, even if the process has not yet reached that step in the process.
		For example, if a process fails at step 3, but the on-fault activity is specified in step 7, if onfault processing is enabled, the process proceeds to the step 7 on-fault rather than halting at step 3.
Start mode	async	Asynchronous initiation is selected by default. Starting business processes asynchronously is recommended. Asynchronous mode is standard Application processing, wherein the business process is placed in queue and processed.
Transaction	False	This option instructs Application to treat the entire process as a single transaction so that either all of the steps complete, or, in the event of an error, none of them do. When an error occurs, no data is committed; data returns to its pre-process state.
		By default, this transaction mode is not enabled.
Queue	4	Application enables you to set performance optimizations by queue, defining queue levels to allocate resources. This number indicates the previously allocated queue level that you want for this business process model for processing.

Parameter	Default	Description
Persistence Level	System Default	The level of data to retain for generating a status report that describes each step that the business process completes. System default indicates that, for the data, configuration is already defined in Application to retain data.
Recovery Level	Manual	The level of recovery for this business process if the business process should halt during execution. Manual requires you to resume or restart the business process manually.
Document Storage Type	System Default	The level of document storage for messages that process when the business process runs. System Default specifies to store messages in the file system or database, according to how you configured archiving and purging in Application.
Life Span	Life Span Days — 2 Life Span Hours — 0 Life Span Type — System Level Removal Method — Archive	The length of time, in days and hours, to retain the data in Application, along with the life span type and removal method.
Complete by Deadline	None Available **Note:** To set a deadline you must change it in the business process.	Complete by – The deadline time, in hours and minutes, by which the business process must complete process once it starts. ◆ First Notification: Hours and Minutes – Whether to receive notification before a business process deadline. ◆ Second Notification: Hours and Minutes – Whether to receive another notification before a business process deadline.
Event Reporting Level	Full	The level of event reporting that is retrieved for this business process when it runs. Full specifies to generate events for the business process, including the business process start and end time, start and end times for all services or services running as a result of this business processes, and any resulting errors and exceptions.

The following BPML code makes up the SWIFTNetClientFA business process:

Note: The **bold** lines indicate information that you need to modify to match your installation.

```
<process name="SWIFTNetClientFA">
   <sequence name="SWIFTNetClientService">
      <operation name="set user token">
         <participant name="SetUserToken"/>
         <output message="SetUserTokenMessage">
            <assign to="USER_TOKEN">admin</assign>
            <assign to="." from="*"/>
         </output>
         <input message="inmsg">
            <assign to="." from="*"/>
         </input>
```

```
        </operation>
        <!-- build SWIFTNET request -->
        <operation>
            <participant name="SWIFTNetClientService"/>
            <output message="handleClientRequest">
                <assign to="." from="*"/>
                <assign to="physicalFilename" from="''"/>
                <assign to="logicalFilename" from="''"/>
                <assign to="transferInfo" from="''"/>
                <assign to="transferDesc" from="''"/>
                <assign to="fileInfo" from="''"/>
                <assign to="fileDesc" from="''"/>
            </output>
            <input message="testing">
                <assign to="." from="*"/>
            </input>
        </operation>
    </sequence>
</process>
```

Before Using the SWIFTNetClientFA Business Process

Before you use the SWIFTNetClientFA business process, you must complete the following tasks:

1. Configure the SWIFTNet Client service (or create a new instance of it) to reflect your installation. See *SWIFTNet Client Service*.

2. Enable Document Tracking in the Business Process Manager.

   Then you can execute the SWIFTNetClientFA business process as part of your SWIFTNet processing.

handleSWIFTNetServerRequest Business Process

The handleSWIFTNetServerRequest business process enables Application to receive SWIFTNet messages. This is the bootstrap business process used by the inbound SWIFTNet request through the SWIFTNet MEFG Server. It is a system business process used by the SWIFTNet Server adapter, which pre-processes the incoming request, search the SWIFTNet Routing Rule table, and route the request payload to the business process for processing.

Note: This business process is used with InterAct processing only.

The handleSWIFTNetServerRequest business process is initiated as part of the following inbound process flow:

1. The SWIFTNet Server adapter invokes the handleSWIFTNetServerRequest business process and passes it all the necessary parameters to send a request.

2. The handleSWIFTNetServerRequest business process invokes the SOAP Inbound service.

3. The SOAP Inbound service:

   ◆ Pre-processes the incoming request.

   ◆ Searches the SWIFTNet Routing Rule table.

   ◆ Routes the request payload to the business process for processing.

4.   Then the SOAP Outbound service invokes the HTTP Response service.

5.   The HTTP Response service sends a response to the request.

This table lists the configuration parameters for the handleSWIFTNetServerRequest business process:

Parameter	Default	Description
Document Tracking	False	When document tracking is enabled for a business process, tracking information is carried with the message throughout the process, and the tracking information is persisted about the message regardless of the persistence level you configured globally for Application.
Set onfault processing	False	Onfault processing allows the process to immediately execute the on-fault activity specified in the process, even if the process has not yet reached that step in the process. For example, if a process fails at step 3, but the on-fault activity is specified in step 7, if onfault processing is enabled, the process proceeds to the step 7 on-fault rather than halting at step 3.
Start mode	async	Asynchronous initiation is selected by default. Starting business processes asynchronously is recommended. Asynchronous mode is standard Application processing, wherein the business process is placed in queue and processed.
Transaction	False	This option instructs Application to treat the entire process as a single transaction so that either all of the steps complete, or, in the event of an error, none of them do. When an error occurs, no data is committed; data returns to its pre-process state. By default, this transaction mode is not enabled.
Queue	4	Application enables you to set performance optimizations by queue, defining queue levels to allocate resources. This number indicates the previously allocated queue level that you want for this business process model for processing.
Persistence Level	System Default	The level of data to retain for generating a status report that describes each step that the business process completes. System default indicates that, for the data, configuration is already defined in Application to retain data.
Recovery Level	Manual	The level of recovery for this business process if the business process should halt during execution. Manual requires you to resume or restart the business process manually.
Document Storage Type	System Default	The level of document storage for messages that process when the business process runs. System Default specifies to store messages in the file system or database, according to how you configured archiving and purging in Application.
Life Span	Life Span Days — 2 Life Span Hours — 0 Life Span Type — System Level Removal Method — Archive	The length of time, in days and hours, to retain the data in Application, along with the life span type and removal method.

Parameter	Default	Description
Complete by Deadline	None Available **Note:** To set a deadline you must change it in the business process.	Complete by – The deadline time, in hours and minutes, by which the business process must complete process once it starts. ◆ First Notification: Hours and Minutes – Whether to receive notification before a business process deadline. ◆ Second Notification: Hours and Minutes – Whether to receive another notification before a business process deadline.
Event Reporting Level	Full	The level of event reporting that is retrieved for this business process when it runs. Full specifies to generate events for the business process, including the business process start and end time, start and end times for all services or services running as a result of this business processes, and any resulting errors and exceptions.

The following BPML code makes up the handleSWIFTNetServerRequest business process:

```
<process name="handleSWIFTNetServerRequest">
    <sequence>
        <operation name="set user token">
            <participant name="SetUserToken"/>
            <output message="SetUserTokenMessage">
                <assign to="USER_TOKEN">admin</assign>
                <assign to="." from="*"/>
            </output>
            <input message="inmsg">
                <assign to="." from="*"/>
            </input>
        </operation>
        <operation name="SoapIn">
            <participant name="SOAPInbound"/>
            <output message="output">
                <assign to="." from="*"/>
                <assign to="bootstrap">false</assign>
                <assign to="SOAP_INTERMEDIATE_NODE">false</assign>
            </output>
            <input message="input">
                <assign to="." from="*"/>
            </input>
        </operation>
        <operation>
            <participant name="SWIFTNetServerAdapter"/>
            <output message="handleServerRequest">
                <assign to="." from="*"/>
            </output>
            <input message="testing">
                <assign to="." from="*"/>
            </input>
        </operation>
        <!-- internal processing by invoking a subprocess -->
        <!-- business-specific processing that will return a response for InterAct -->
```

```
    <operation>
        <participant name="InvokeSubProcessService"/>
        <output message="Xout">
            <assign to="INVOKE_MODE">SYNC</assign>
            <assign to="." from="*"/>
        </output>
        <input message="Xin">
            <assign to="." from="*"/>
        </input>
    </operation>
    <!-- this is to construct the server response message back to GIS Server
application -->
    <operation>
        <participant name="SWIFTNetServerAdapter"/>
        <output message="handleServerResponse">
            <assign to="." from="*"/>
            <assign to="interface" from="SwiftServerRequest/interface/text()"/>
            <assign to="messageID" from="SwiftServerRequest/messageID/text()"/>
            <assign to="Status">Accepted</assign>
            <assign to="deliveryNotification"
from="SwiftServerRequest/deliveryNotification/text()"/>
            <assign to="SnF" from="SwiftServerRequest/SnF/text()"/>
        </output>
        <input message="testing">
            <assign to="." from="*"/>
        </input>
    </operation>
    <operation name="SoapOut">
        <participant name="SOAPOutbound"/>
        <output message="output">
            <assign to="." from="*"/>
            <assign to="SOAP_MODE">respond</assign>
        </output>
        <input message="input">
            <assign to="." from="*"/>
        </input>
    </operation>
    <assign to="doc-has-headers">true</assign>
    <operation name="HttpResponse">
        <participant name="HttpRespond"/>
        <output message="Xout">
            <assign to="." from="*"/>
        </output>
        <input message="Xin">
            <assign to="." from="*"/>
        </input>
    </operation>
    <onFault>
        <!-- On Fault, we will clear PrimDoc, construct Rejected response and
soap-envelope it -->
        <sequence>
            <operation name="ReleasePrimDoc">
                <participant name="ReleaseService"/>
                <output message="outmsg">
                    <assign to="TARGET">/ProcessData/PrimaryDocument</assign>
                    <assign to="." from="*"/>
```

```
                        </output>
                        <input message="inmsg"/>
                    </operation>
                    <operation>
                        <participant name="SWIFTNetServerAdapter"/>
                        <output message="handleServerResponse">
                            <assign to="." from="*"/>
                            <assign to="interface"
from="SwiftServerRequest/interface/text()"/>
                            <assign to="messageID"
from="SwiftServerRequest/messageID/text()"/>
                            <assign to="Status">Rejected</assign>
                            <assign to="Description">Unable to get the Server
Response</assign>
                            <assign to="Info">Failure in getting the Server Response</assign>
                            <assign to="deliveryNotification"
from="SwiftServerRequest/deliveryNotification/text()"/>
                            <assign to="SnF" from="SwiftServerRequest/SnF/text()"/>
                        </output>
                        <input message="testing">
                            <assign to="." from="*"/>
                        </input>
                    </operation>
                    <operation name="SoapOut">
                        <participant name="SOAPOutbound"/>
                        <output message="output">
                            <assign to="." from="*"/>
                            <assign to="SOAP_MODE">respond</assign>
                        </output>
                        <input message="input">
                            <assign to="." from="*"/>
                        </input>
                    </operation>
                    <assign to="doc-has-headers">true</assign>
                    <operation name="HttpResponse">
                        <participant name="HttpRespond"/>
                        <output message="Xout">
                            <assign to="." from="*"/>
                        </output>
                        <input message="Xin">
                            <assign to="." from="*"/>
                        </input>
                    </operation>
                </sequence>
            </onFault>
        </sequence>
</process>
```

Before Using the handleSWIFTNetServerRequest Business Process

Before you use the handleSWIFTNetServerRequest business process, you must complete the following tasks:

1. Configure the SWIFTNet Server adapter to reflect your installation. See *SWIFTNet Server Adapter.*

2. Enable Document Tracking in the Business Process Manager for the handleSWIFTNetServerRequest business process.

handleSWIFTNetServerSnFRequest Business Process

The handleSWIFTNetServerSnFRequest business process enables Application to receive SWIFTNet store-and-forward messages. This is a bootstrap business process used by the inbound SWIFTNet request that includes a store-and-forward option. For the store-and-forward option, an incoming request is not processed immediately, but instead is stored in the responder mailbox in Application to be responded to later. Application then sends an acknowledgement that the request has been successfully stored to the requestor through the SWIFTNet MEFG Server.

Note: This business process is used with InterAct processing only.

The handleSWIFTNetServerSnFRequest business process is initiated as part of the following inbound process flow:

1. The SWIFTNet Server adapter invokes the handleSWIFTNetServerSnFRequest business process and passes it all the necessary parameters to send a request.

2. The handleSWIFTNetServerSnFRequest business process invokes the SOAP Inbound service.

3. The SOAP Inbound service:

    ◆   Pre-processes the incoming request.

    ◆   Searches the SWIFTNet Routing Rule table.

4. The Mailbox Add service Routes the request payload to the responder mailbox to be responded to later.

5. Then the SOAP Outbound service invokes the HTTP Response service.

6. The HTTP Response service sends an acknowledgement to the requestor that the request has been successfully stored.

This table lists the configuration parameters for the handleSWIFTNetServerSnFRequest business process:

Parameter	Default	Description
Document Tracking	False	When document tracking is enabled for a business process, tracking information is carried with the message throughout the process, and the tracking information is persisted about the message regardless of the persistence level you configured globally for Application.
Set onfault processing	False	Onfault processing allows the process to immediately execute the on-fault activity specified in the process, even if the process has not yet reached that step in the process.
		For example, if a process fails at step 3, but the on-fault activity is specified in step 7, if onfault processing is enabled, the process proceeds to the step 7 on-fault rather than halting at step 3.
Start mode	async	Asynchronous initiation is selected by default. Starting business processes asynchronously is recommended. Asynchronous mode is standard Application processing, wherein the business process is placed in queue and processed.
Transaction	False	This option instructs Application to treat the entire process as a single transaction so that either all of the steps complete, or, in the event of an error, none of them do. When an error occurs, no data is committed; data returns to its pre-process state.
		By default, this transaction mode is not enabled.
Queue	4	Application enables you to set performance optimizations by queue, defining queue levels to allocate resources. This number indicates the previously allocated queue level that you want for this business process model for processing.
Persistence Level	System Default	The level of data to retain for generating a status report that describes each step that the business process completes. System default indicates that, for the data, configuration is already defined in Application to retain data.
Recovery Level	Manual	The level of recovery for this business process if the business process should halt during execution. Manual requires you to resume or restart the business process manually.
Document Storage Type	System Default	The level of document storage for messages that process when the business process runs. System Default specifies to store messages in the file system or database, according to how you configured archiving and purging in Application.
Life Span	Life Span Days — 2 Life Span Hours — 0 Life Span Type — System Level Removal Method — Archive	The length of time, in days and hours, to retain the data in Application, along with the life span type and removal method.

Parameter	Default	Description
Complete by Deadline	None Available **Note:** To set a deadline you must change it in the business process.	Complete by – The deadline time, in hours and minutes, by which the business process must complete process once it starts. ◆ First Notification: Hours and Minutes – Whether to receive notification before a business process deadline. ◆ Second Notification: Hours and Minutes – Whether to receive another notification before a business process deadline.
Event Reporting Level	Full	The level of event reporting that is retrieved for this business process when it runs. Full specifies to generate events for the business process, including the business process start and end time, start and end times for all services or services running as a result of this business processes, and any resulting errors and exceptions.

The following BPML code makes up the handleSWIFTNetServerSnFRequest business process:

```
<process name="handleSWIFTNetServerSnFRequest">
   <rule name="not_DeliveryNotificationRequest">
      <condition>SwiftServerRequest/deliveryNotification = 'FALSE'</condition>
   </rule>
   <sequence>
      <operation name="set user token">
         <participant name="SetUserToken"/>
         <output message="SetUserTokenMessage">
            <assign to="USER_TOKEN">admin</assign>
            <assign to="." from="*"/>
         </output>
         <input message="inmsg">
            <assign to="." from="*"/>
         </input>
      </operation>
      <operation name="SoapIn">
         <participant name="SOAPInbound"/>
         <output message="output">
            <assign to="." from="*"/>
            <assign to="bootstrap">false</assign>
            <assign to="SOAP_INTERMEDIATE_NODE">false</assign>
         </output>
         <input message="input">
            <assign to="." from="*"/>
         </input>
      </operation>
      <operation>
         <participant name="SWIFTNetServerAdapter"/>
         <output message="handleServerRequest">
            <assign to="." from="*"/>
         </output>
         <input message="testing">
            <assign to="." from="*"/>
         </input>
```

```
        </operation>
        <!-- internal processing for SnF is to put into a Mailbox so that it can
bootstrap internal business process later-->
        <!-- Mailbox path is based on SwiftServerRequest/responderDN/requestorDN/for
InterAct -->
        <choice>
            <select>
                <case ref="not_DeliveryNotificationRequest"
activity="not_DeliveryNotificationRequest"/>
            </select>
            <sequence name="not_DeliveryNotificationRequest">
                <operation name="Mailbox Add Service">
                    <participant name="MailboxAdd"/>
                    <output message="AddRequest">
                        <assign to="." from="*"/>
                        <assign to="MailboxPath" from="concat('/',
SwiftServerRequest/responderDN/text(),'/',SwiftServerRequest/requestorDN/text())"/>
                        <assign to="ContentType">ascii</assign>
                    </output>
                    <input message="inmsg">
                        <assign to="AddResults" from="*"/>
                    </input>
                </operation>
            </sequence>
        </choice>
        <operation>
            <participant name="SWIFTNetServerAdapter"/>
            <output message="handleServerResponse">
                <assign to="." from="*"/>
                <assign to="interface" from="SwiftServerRequest/interface/text()"/>
                <assign to="messageID" from="SwiftServerRequest/messageID/text()"/>
                <assign to="Status">Accepted</assign>
                <assign to="deliveryNotification"
from="SwiftServerRequest/deliveryNotification/text()"/>
                <assign to="SnF" from="SwiftServerRequest/SnF/text()"/>
            </output>
            <input message="testing">
                <assign to="." from="*"/>
            </input>
        </operation>
        <operation name="SoapOut">
            <participant name="SOAPOutbound"/>
            <output message="output">
                <assign to="." from="*"/>
                <assign to="SOAP_MODE">respond</assign>
            </output>
            <input message="input">
                <assign to="." from="*"/>
            </input>
        </operation>
        <assign to="doc-has-headers">true</assign>
        <operation name="HttpResponse">
            <participant name="HttpRespond"/>
            <output message="Xout">
                <assign to="." from="*"/>
            </output>
```

```
            <input message="Xin">
                <assign to="." from="*"/>
            </input>
        </operation>
    <onFault>
        <sequence>
            <operation name="ReleasePrimDoc">
                <participant name="ReleaseService"/>
                <output message="outmsg">
                    <assign to="TARGET">/ProcessData/PrimaryDocument</assign>
                    <assign to="." from="*"/>
                </output>
                <input message="inmsg"/>
            </operation>
            <operation>
                <participant name="SWIFTNetServerAdapter"/>
                <output message="handleServerResponse">
                    <assign to="." from="*"/>
                    <assign to="interface"
from="SwiftServerRequest/interface/text()"/>
                    <assign to="messageID"
from="SwiftServerRequest/messageID/text()"/>
                    <assign to="Status">Rejected</assign>
                    <assign to="Description">Unable to get the Server
Response</assign>
                    <assign to="Info">Failure in getting the Server Response</assign>
                    <assign to="deliveryNotification"
from="SwiftServerRequest/deliveryNotification/text()"/>
                    <assign to="SnF" from="SwiftServerRequest/SnF/text()"/>
                </output>
                <input message="testing">
                    <assign to="." from="*"/>
                </input>
            </operation>
            <operation name="SoapOut">
                <participant name="SOAPOutbound"/>
                <output message="output">
                    <assign to="." from="*"/>
                    <assign to="SOAP_MODE">respond</assign>
                </output>
                <input message="input">
                    <assign to="." from="*"/>
                </input>
            </operation>
            <assign to="doc-has-headers">true</assign>
            <operation name="HttpResponse">
                <participant name="HttpRespond"/>
                <output message="Xout">
                    <assign to="." from="*"/>
                </output>
                <input message="Xin">
                    <assign to="." from="*"/>
                </input>
            </operation>
        </sequence>
    </onFault>
```

```
    </sequence>
</process>
```

Before Using the handleSWIFTNetServerSnFRequest Business Process

Before you use the handleSWIFTNetServerSnFRequest business process, you must complete the following tasks:

1. Configure the SWIFTNet Server adapter to reflect your installation. See *SWIFTNet Server Adapter.*

2. Enable Document Tracking in the Business Process Manager for the handleSWIFTNetServerSnFRequest business process.

handleSWIFTNetServerFARequest Business Process

The handleSWIFTNetServerFARequest business process enables Application to receive SWIFTNet messages. This is the bootstrap business process used by the inbound SWIFTNet request through the SWIFTNet MEFG Server. It is a system business process used by the SWIFTNet Server adapter, which pre-processes the incoming request, search the SWIFTNet Routing Rule table, and route the request payload to the business process for processing.

Note: This business process is used with InterAct processing only.

The handleSWIFTNetServerFARequest business process is initiated as part of the following inbound process flow:

1. The SWIFTNet Server adapter invokes the handleSWIFTNetServerFARequest business process and passes it all the necessary parameters to send a request.

2. The handleSWIFTNetServerRequest business process invokes the SOAP Inbound service.

3. The SOAP Inbound service:

    ◆ Pre-processes the incoming request.

    ◆ Searches the SWIFTNet Routing Rule table.

    ◆ Routes the request payload to the business process for processing.

4. Then the SOAP Outbound service invokes the HTTP Response service.

5. The HTTP Response service sends a response to the request.

This table lists the configuration parameters for the handleSWIFTNetServerFARequest business process:

Parameter	Default	Description
Document Tracking	False	When document tracking is enabled for a business process, tracking information is carried with the message throughout the process, and the tracking information is persisted about the message regardless of the persistence level you configured globally for Application.
Set onfault processing	False	Onfault processing allows the process to immediately execute the on-fault activity specified in the process, even if the process has not yet reached that step in the process. For example, if a process fails at step 3, but the on-fault activity is specified in step 7, if onfault processing is enabled, the process proceeds to the step 7 on-fault rather than halting at step 3.
Start mode	async	Asynchronous initiation is selected by default. Starting business processes asynchronously is recommended. Asynchronous mode is standard Application processing, wherein the business process is placed in queue and processed.
Transaction	False	This option instructs Application to treat the entire process as a single transaction so that either all of the steps complete, or, in the event of an error, none of them do. When an error occurs, no data is committed; data returns to its pre-process state. By default, this transaction mode is not enabled.
Queue	4	Application enables you to set performance optimizations by queue, defining queue levels to allocate resources. This number indicates the previously allocated queue level that you want for this business process model for processing.
Persistence Level	System Default	The level of data to retain for generating a status report that describes each step that the business process completes. System default indicates that, for the data, configuration is already defined in Application to retain data.
Recovery Level	Manual	The level of recovery for this business process if the business process should halt during execution. Manual requires you to resume or restart the business process manually.
Document Storage Type	System Default	The level of document storage for messages that process when the business process runs. System Default specifies to store messages in the file system or database, according to how you configured archiving and purging in Application.
Life Span	Life Span Days — 2 Life Span Hours — 0 Life Span Type — System Level Removal Method — Archive	The length of time, in days and hours, to retain the data in Application, along with the life span type and removal method.

Parameter	Default	Description
Complete by Deadline	None Available **Note:** To set a deadline you must change it in the business process.	Complete by – The deadline time, in hours and minutes, by which the business process must complete process once it starts. ◆ First Notification: Hours and Minutes – Whether to receive notification before a business process deadline. ◆ Second Notification: Hours and Minutes – Whether to receive another notification before a business process deadline.
Event Reporting Level	Full	The level of event reporting that is retrieved for this business process when it runs. Full specifies to generate events for the business process, including the business process start and end time, start and end times for all services or services running as a result of this business processes, and any resulting errors and exceptions.

The following BPML code makes up the handleSWIFTNetServerFARequest business process:

```
<process name="handleSWIFTNetServerFARequest">
    <sequence>
        <operation name="set user token">
            <participant name="SetUserToken"/>
            <output message="SetUserTokenMessage">
                <assign to="USER_TOKEN">admin</assign>
                <assign to="." from="*"/>
            </output>
            <input message="inmsg">
                <assign to="." from="*"/>
            </input>
        </operation>
        <operation name="SoapIn">
            <participant name="SOAPInbound"/>
            <output message="output">
                <assign to="." from="*"/>
                <assign to="bootstrap">false</assign>
                <assign to="SOAP_INTERMEDIATE_NODE">false</assign>
            </output>
            <input message="input">
                <assign to="." from="*"/>
            </input>
        </operation>
        <operation>
            <participant name="SWIFTNetServerAdapter"/>
            <output message="handleServerRequest">
                <assign to="." from="*"/>
            </output>
            <input message="testing">
                <assign to="." from="*"/>
            </input>
        </operation>
        <!-- this is to construct the server response message back to GIS Server
application -->
```

```
        <operation>
            <participant name="SWIFTNetServerAdapter"/>
            <output message="handleServerResponse">
                <assign to="." from="*"/>
                <assign to="interfaceMode"
from="SwiftServerRequest/interfaceMode/text()"/>
                <assign to="messageID" from="SwiftServerRequest/messageID/text()"/>
                <assign to="Status">Accepted</assign>
                <assign to="deliveryNotification"
from="SwiftServerRequest/deliveryNotification/text()"/>
                <assign to="SnF" from="SwiftServerRequest/SnF/text()"/>
            </output>
            <input message="testing">
                <assign to="." from="*"/>
            </input>
        </operation>
        <operation name="SoapOut">
            <participant name="SOAPOutbound"/>
            <output message="output">
                <assign to="." from="*"/>
                <assign to="SOAP_MODE">respond</assign>
            </output>
            <input message="input">
                <assign to="." from="*"/>
            </input>
        </operation>
        <assign to="doc-has-headers">true</assign>
        <operation name="HttpResponse">
            <participant name="HttpRespond"/>
            <output message="Xout">
                <assign to="." from="*"/>
            </output>
            <input message="Xin">
                <assign to="." from="*"/>
            </input>
        </operation>
        <onFault>
            <!-- On Fault, we will clear PrimDoc, construct Rejected response and
soap-envelope it -->
            <sequence>
                <operation name="ReleasePrimDoc">
                    <participant name="ReleaseService"/>
                    <output message="outmsg">
                        <assign to="TARGET">/ProcessData/PrimaryDocument</assign>
                        <assign to="." from="*"/>
                    </output>
                    <input message="inmsg"/>
                </operation>
                <operation>
                    <participant name="SWIFTNetServerAdapter"/>
                    <output message="handleServerResponse">
                        <assign to="." from="*"/>
                        <assign to="interfaceMode"
from="SwiftServerRequest/interfaceMode/text()"/>
                        <assign to="messageID"
from="SwiftServerRequest/messageID/text()"/>
```

```
                        <assign to="Status">Rejected</assign>
                        <assign to="Description">Unable to get the Server
Response</assign>
                        <assign to="Info">Failure in getting the Server Response</assign>
                        <assign to="deliveryNotification"
from="SwiftServerRequest/deliveryNotification/text()"/>
                        <assign to="SnF" from="SwiftServerRequest/SnF/text()"/>
                    </output>
                    <input message="testing">
                        <assign to="." from="*"/>
                    </input>
                </operation>
                <operation name="SoapOut">
                    <participant name="SOAPOutbound"/>
                    <output message="output">
                        <assign to="." from="*"/>
                        <assign to="SOAP_MODE">respond</assign>
                    </output>
                    <input message="input">
                        <assign to="." from="*"/>
                    </input>
                </operation>
                <assign to="doc-has-headers">true</assign>
                <operation name="HttpResponse">
                    <participant name="HttpRespond"/>
                    <output message="Xout">
                        <assign to="." from="*"/>
                    </output>
                    <input message="Xin">
                        <assign to="." from="*"/>
                    </input>
                </operation>
            </sequence>
        </onFault>
    </sequence>
</process>
```

Before Using the handleSWIFTNetServerFARequest Business Process

Before you use the handleSWIFTNetServerFARequest business process, you must complete the following tasks:

1. Configure the SWIFTNet Server adapter to reflect your installation. See *SWIFTNet Server Adapter.*

2. Enable Document Tracking in the Business Process Manager for the handleSWIFTNetServerFARequest business process.

handleSWIFTNetServerFASnFRequest Business Process

The handleSWIFTNetServerFASnFRequest business process enables Application to receive SWIFTNet store-and-forward messages. This is a bootstrap business process used by the inbound SWIFTNet request that includes a store-and-forward option. For the store-and-forward option, an incoming request is not processed immediately, but instead is stored in the responder mailbox in Application to be responded to

later. Application then sends an acknowledgement that the request has been successfully stored to the requestor through the SWIFTNet MEFG Server.

Note: This business process is used for FileAct processing only.

The handleSWIFTNetServerFASnFRequest business process is initiated as part of the following inbound process flow:

1. The SWIFTNet Server adapter invokes the handleSWIFTNetServerFASnFRequest business process and passes it all the necessary parameters to send a request.

2. The handleSWIFTNetServerFASnFRequest business process invokes the SOAP Inbound service.

3. The SOAP Inbound service:

   ◆ Pre-processes the incoming request.

   ◆ Searches the SWIFTNet Routing Rule table.

4. The Mailbox Add service Routes the request payload to the responder mailbox to be responded to later.

5. Then the SOAP Outbound service invokes the HTTP Response service.

6. The HTTP Response service sends an acknowledgement to the requestor that the request has been successfully stored.

This table lists the configuration parameters for the handleSWIFTNetServerFASnFRequest business process:

Parameter	Default	Description
Document Tracking	False	When document tracking is enabled for a business process, tracking information is carried with the message throughout the process, and the tracking information is persisted about the message regardless of the persistence level you configured globally for Application.
Set onfault processing	False	Onfault processing allows the process to immediately execute the on-fault activity specified in the process, even if the process has not yet reached that step in the process. For example, if a process fails at step 3, but the on-fault activity is specified in step 7, if onfault processing is enabled, the process proceeds to the step 7 on-fault rather than halting at step 3.
Start mode	async	Asynchronous initiation is selected by default. Starting business processes asynchronously is recommended. Asynchronous mode is standard Application processing, wherein the business process is placed in queue and processed.
Transaction	False	This option instructs Application to treat the entire process as a single transaction so that either all of the steps complete, or, in the event of an error, none of them do. When an error occurs, no data is committed; data returns to its pre-process state. By default, this transaction mode is not enabled.
Queue	4	Application enables you to set performance optimizations by queue, defining queue levels to allocate resources. This number indicates the previously allocated queue level that you want for this business process model for processing.
Persistence Level	System Default	The level of data to retain for generating a status report that describes each step that the business process completes. System default indicates that, for the data, configuration is already defined in Application to retain data.
Recovery Level	Manual	The level of recovery for this business process if the business process should halt during execution. Manual requires you to resume or restart the business process manually.
Document Storage Type	System Default	The level of document storage for messages that process when the business process runs. System Default specifies to store messages in the file system or database, according to how you configured archiving and purging in Application.

Parameter	Default	Description
Life Span	Life Span Days — 2 Life Span Hours — 0 Life Span Type — System Level Removal Method — Archive	The length of time, in days and hours, to retain the data in Application, along with the life span type and removal method.
Complete by Deadline	None Available **Note:** To set a deadline you must change it in the business process.	Complete by – The deadline time, in hours and minutes, by which the business process must complete process once it starts. ♦ First Notification: Hours and Minutes – Whether to receive notification before a business process deadline. ♦ Second Notification: Hours and Minutes – Whether to receive another notification before a business process deadline.
Event Reporting Level	Full	The level of event reporting that is retrieved for this business process when it runs. Full specifies to generate events for the business process, including the business process start and end time, start and end times for all services or services running as a result of this business processes, and any resulting errors and exceptions.

The following BPML code makes up the handleSWIFTNetServerFASnFRequest business process:

```
<process name="handleSWIFTNetServerFASnFRequest">
   <rule name="not_DeliveryNotificationRequest">
      <condition>SwiftServerRequest/deliveryNotification = 'FALSE'</condition>
   </rule>
   <sequence>
      <operation name="set user token">
         <participant name="SetUserToken"/>
         <output message="SetUserTokenMessage">
            <assign to="USER_TOKEN">admin</assign>
            <assign to="." from="*"/>
         </output>
         <input message="inmsg">
            <assign to="." from="*"/>
         </input>
      </operation>
      <operation name="SoapIn">
         <participant name="SOAPInbound"/>
         <output message="output">
            <assign to="." from="*"/>
            <assign to="bootstrap">false</assign>
            <assign to="SOAP_INTERMEDIATE_NODE">false</assign>
         </output>
         <input message="input">
            <assign to="." from="*"/>
         </input>
      </operation>
```

```
        <operation>
            <participant name="SWIFTNetServerAdapter"/>
            <output message="handleServerRequest">
               <assign to="." from="*"/>
            </output>
            <input message="testing">
               <assign to="." from="*"/>
            </input>
        </operation>
        <operation>
            <participant name="SWIFTNetServerAdapter"/>
            <output message="handleServerResponse">
               <assign to="." from="*"/>
               <assign to="interfaceMode"
from="SwiftServerRequest/interfaceMode/text()"/>
               <assign to="messageID" from="SwiftServerRequest/messageID/text()"/>
               <assign to="Status">Accepted</assign>
               <assign to="deliveryNotification"
from="SwiftServerRequest/deliveryNotification/text()"/>
               <assign to="SnF" from="SwiftServerRequest/SnF/text()"/>
            </output>
            <input message="testing">
               <assign to="." from="*"/>
            </input>
        </operation>
        <operation name="SoapOut">
            <participant name="SOAPOutbound"/>
            <output message="output">
               <assign to="." from="*"/>
               <assign to="SOAP_MODE">respond</assign>
            </output>
            <input message="input">
               <assign to="." from="*"/>
            </input>
        </operation>
        <assign to="doc-has-headers">true</assign>
        <operation name="HttpResponse">
            <participant name="HttpRespond"/>
            <output message="Xout">
               <assign to="." from="*"/>
            </output>
            <input message="Xin">
               <assign to="." from="*"/>
            </input>
        </operation>
        <onFault>
            <sequence>
               <operation name="ReleasePrimDoc">
                  <participant name="ReleaseService"/>
                  <output message="outmsg">
                     <assign to="TARGET">/ProcessData/PrimaryDocument</assign>
                     <assign to="." from="*"/>
                  </output>
                  <input message="inmsg"/>
               </operation>
               <operation>
```

```
                    <participant name="SWIFTNetServerAdapter"/>
                    <output message="handleServerResponse">
                       <assign to="." from="*"/>
                       <assign to="interfaceMode"
from="SwiftServerRequest/interfaceMode/text()"/>
                       <assign to="messageID"
from="SwiftServerRequest/messageID/text()"/>
                       <assign to="Status">Rejected</assign>
                       <assign to="Description">Unable to get the Server
Response</assign>
                       <assign to="Info">Failure in getting the Server Response</assign>
                       <assign to="deliveryNotification"
from="SwiftServerRequest/deliveryNotification/text()"/>
                       <assign to="SnF" from="SwiftServerRequest/SnF/text()"/>
                    </output>
                    <input message="testing">
                       <assign to="." from="*"/>
                    </input>
                 </operation>
                 <operation name="SoapOut">
                    <participant name="SOAPOutbound"/>
                    <output message="output">
                       <assign to="." from="*"/>
                       <assign to="SOAP_MODE">respond</assign>
                    </output>
                    <input message="input">
                       <assign to="." from="*"/>
                    </input>
                 </operation>
                 <assign to="doc-has-headers">true</assign>
                 <operation name="HttpResponse">
                    <participant name="HttpRespond"/>
                    <output message="Xout">
                       <assign to="." from="*"/>
                    </output>
                    <input message="Xin">
                       <assign to="." from="*"/>
                    </input>
                 </operation>
              </sequence>
           </onFault>
       </sequence>
</process>
```

Before Using the handleSWIFTNetServerFASnFRequest Business Process

Before you use the handleSWIFTNetServerFASnFRequest business process, you must complete the
following tasks:

1. Configure the SWIFTNet Server adapter to reflect your installation. See *SWIFTNet Server Adapter*.

2. Enable Document Tracking in the Business Process Manager for the
   handleSWIFTNetServerFASnFRequest business process.

handleSWIFTNetServerFAEvent Business Process

The handleSWIFTNetServerFAEvent business process is used by the SWIFTNet MEFG Server. It is a system business process called by the SWIFTNet Server adapter that preprocesses the incoming FileAct Event with a COMPLETED status, searches the SWIFTNet routing rule table and bootstraps the business process for processing.

Note: This business process is used for FileAct processing only.

The handleSWIFTNetServerFAEvent business process is initiated as part of the following inbound process flow:

1. The SWIFTNet Server adapter invokes the handleSWIFTNetServerFAEvent business process and passes it all the necessary parameters to send a request.

2. The handleSWIFTNetServerFAEvent business process preprocesses the incoming FileAct event.

3. The handleSWIFTNetServerFAEvent business process searches the SWIFTNet Routing Rule table.

4. Then the business process bootstraps the appropriate business process for processing.

This table lists the configuration parameters for the handleSWIFTNetServerFAEvent business process:

Parameter	Default	Description
Document Tracking	False	When document tracking is enabled for a business process, tracking information is carried with the message throughout the process, and the tracking information is persisted about the message regardless of the persistence level you configured globally for Application.
Set onfault processing	False	Onfault processing allows the process to immediately execute the on-fault activity specified in the process, even if the process has not yet reached that step in the process.
		For example, if a process fails at step 3, but the on-fault activity is specified in step 7, if onfault processing is enabled, the process proceeds to the step 7 on-fault rather than halting at step 3.
Start mode	async	Asynchronous initiation is selected by default. Starting business processes asynchronously is recommended. Asynchronous mode is standard Application processing, wherein the business process is placed in queue and processed.
Transaction	False	This option instructs Application to treat the entire process as a single transaction so that either all of the steps complete, or, in the event of an error, none of them do. When an error occurs, no data is committed; data returns to its pre-process state.
		By default, this transaction mode is not enabled.
Queue	4	Application enables you to set performance optimizations by queue, defining queue levels to allocate resources. This number indicates the previously allocated queue level that you want for this business process model for processing.
Persistence Level	System Default	The level of data to retain for generating a status report that describes each step that the business process completes. System default indicates that, for the data, configuration is already defined in Application to retain data.
Recovery Level	Manual	The level of recovery for this business process if the business process should halt during execution. Manual requires you to resume or restart the business process manually.
Document Storage Type	System Default	The level of document storage for messages that process when the business process runs. System Default specifies to store messages in the file system or database, according to how you configured archiving and purging in Application.
Life Span	Life Span Days — 2 Life Span Hours — 0 Life Span Type — System Level Removal Method — Archive	The length of time, in days and hours, to retain the data in Application, along with the life span type and removal method.

Parameter	Default	Description
Complete by Deadline	None Available **Note:** To set a deadline you must change it in the business process.	Complete by – The deadline time, in hours and minutes, by which the business process must complete process once it starts. ◆ First Notification: Hours and Minutes – Whether to receive notification before a business process deadline. ◆ Second Notification: Hours and Minutes – Whether to receive another notification before a business process deadline.
Event Reporting Level	Full	The level of event reporting that is retrieved for this business process when it runs. Full specifies to generate events for the business process, including the business process start and end time, start and end times for all services or services running as a result of this business processes, and any resulting errors and exceptions.

The following BPML code makes up the handleSWIFTNetServerFAEvent business process:

```
<process name="handleSWIFTNetServerFAEvent">
    <rule name="Backend_Workflow">
        <condition>DoFABackendProcess= 'workflow'</condition>
    </rule>
    <rule name="Backend_MailBox">
        <condition>DoFABackendProcess= 'mailbox'</condition>
    </rule>
    <sequence>
        <operation name="set user token">
            <participant name="SetUserToken"/>
            <output message="SetUserTokenMessage">
               <assign to="USER_TOKEN">admin</assign>
               <assign to="." from="*"/>
            </output>
            <input message="inmsg">
               <assign to="." from="*"/>
            </input>
        </operation>
        <operation name="SoapIn">
            <participant name="SOAPInbound"/>
            <output message="output">
               <assign to="." from="*"/>
               <assign to="bootstrap">false</assign>
               <assign to="SOAP_INTERMEDIATE_NODE">false</assign>
            </output>
            <input message="input">
               <assign to="." from="*"/>
            </input>
        </operation>
        <operation>
            <participant name="SWIFTNetServerAdapter"/>
            <output message="handleServerRequest">
               <assign to="." from="*"/>
            </output>
```

```
            <input message="testing">
               <assign to="." from="*"/>
            </input>
        </operation>
        <!-- Process Completed FA Event with incoming, put or get request-->
        <!-- non SnF - bootstrap Workflow -->
        <!-- SnF - put in MailBox -->
        <choice>
            <select>
               <case ref="Backend_Workflow" activity="processBackend_Workflow"/>
               <case ref="Backend_MailBox" activity="processBackend_MailBox"/>
            </select>
            <sequence name="processBackend_Workflow">
               <operation>
                   <participant name="InvokeSubProcessService"/>
                   <output message="Xout">
                      <assign to="INVOKE_MODE">SYNC</assign>
                      <assign to="." from="*"/>
                   </output>
                   <input message="Xin">
                      <assign to="." from="*"/>
                   </input>
               </operation>
            </sequence>
            <sequence name="processBackend_MailBox">
               <operation name="Mailbox Add Service">
                   <participant name="MailboxAdd"/>
                   <output message="AddRequest">
                      <assign to="." from="*"/>
                      <assign to="MailboxPath" from="concat('/',
SwiftServerRequest/responderDN/text(),'/',SwiftServerRequest/requestorDN/text())"/>
                      <assign to="ContentType">ascii</assign>
                   </output>
                   <input message="inmsg">
                      <assign to="AddResults" from="*"/>
                   </input>
               </operation>
            </sequence>
        </choice>
        <!-- this is to construct the server response message back to GIS Server
application -->
        <operation>
            <participant name="SWIFTNetServerAdapter"/>
            <output message="handleServerResponse">
               <assign to="." from="*"/>
               <assign to="interfaceMode"
from="SwiftServerRequest/interfaceMode/text()"/>
               <assign to="messageID" from="SwiftServerRequest/messageID/text()"/>
               <assign to="Status">Accepted</assign>
               <assign to="deliveryNotification"
from="SwiftServerRequest/deliveryNotification/text()"/>
               <assign to="SnF" from="SwiftServerRequest/SnF/text()"/>
            </output>
            <input message="testing">
               <assign to="." from="*"/>
            </input>
```

```
        </operation>
        <operation name="SoapOut">
            <participant name="SOAPOutbound"/>
            <output message="output">
               <assign to="." from="*"/>
               <assign to="SOAP_MODE">respond</assign>
            </output>
            <input message="input">
               <assign to="." from="*"/>
            </input>
        </operation>
        <assign to="doc-has-headers">true</assign>
        <operation name="HttpResponse">
            <participant name="HttpRespond"/>
            <output message="Xout">
               <assign to="." from="*"/>
            </output>
            <input message="Xin">
               <assign to="." from="*"/>
            </input>
        </operation>
        <onFault>
            <!-- On Fault, we will clear PrimDoc, construct Rejected response and
soap-envelope it -->
            <sequence>
                <operation name="ReleasePrimDoc">
                    <participant name="ReleaseService"/>
                    <output message="outmsg">
                       <assign to="TARGET">/ProcessData/PrimaryDocument</assign>
                       <assign to="." from="*"/>
                    </output>
                    <input message="inmsg"/>
                </operation>
                <operation>
                    <participant name="SWIFTNetServerAdapter"/>
                    <output message="handleServerResponse">
                       <assign to="." from="*"/>
                       <assign to="interfaceMode"
from="SwiftServerRequest/interfaceMode/text()"/>
                       <assign to="messageID"
from="SwiftServerRequest/messageID/text()"/>
                       <assign to="Status">Rejected</assign>
                       <assign to="Description">Fail in Backend Process and unable to
get the Server Response</assign>
                       <assign to="Info">Failure in Backend Process and getting the
Server Response</assign>
                       <assign to="deliveryNotification"
from="SwiftServerRequest/deliveryNotification/text()"/>
                       <assign to="SnF" from="SwiftServerRequest/SnF/text()"/>
                    </output>
                    <input message="testing">
                       <assign to="." from="*"/>
                    </input>
                </operation>
                <operation name="SoapOut">
                    <participant name="SOAPOutbound"/>
```

```
                    <output message="output">
                        <assign to="." from="*"/>
                        <assign to="SOAP_MODE">respond</assign>
                    </output>
                    <input message="input">
                        <assign to="." from="*"/>
                    </input>
                </operation>
                <assign to="doc-has-headers">true</assign>
                <operation name="HttpResponse">
                    <participant name="HttpRespond"/>
                    <output message="Xout">
                        <assign to="." from="*"/>
                    </output>
                    <input message="Xin">
                        <assign to="." from="*"/>
                    </input>
                </operation>
            </sequence>
        </onFault>
    </sequence>
</process>
```

Before Using the handleSWIFTNetServerFAEvent Business Process

Before you use the handleSWIFTNetServerFAEvent business process, you must complete the following tasks:

1. Configure the SWIFTNet Server adapter to reflect your installation. See *SWIFTNet Server Adapter.*

2. Enable Document Tracking in the Business Process Manager for the handleSWIFTNetServerFAEvent business process.

SWIFTMessageEntryEnvelope

The SWIFTMessageEntryEnvelope business process is the default business process used to look up the correct envelope, based on the information provided by the user while configuring the Send information in the SWIFT Message Entry Workstation.

The SWIFTMessageEntryEnvelope business process is initiated as part of the following outbound process flow:

1. Create a SWIFT message using the SWIFT Message Entry Workstation.

2. Validate the message and mark it **Ready to Send**.

3. Do not change the default business process and send the SWIFT message you created.

4. The SWIFTMessageEntryEnvelope business process invokes the EDI Encoder service.

5. The EDI Encoder service includes the Generic Envelope service.

6. The Generic Envelope service invokes the business process configured in the SWIFT Outbound Envelope to send the SWIFT message.

This table lists the configuration parameters for the SWIFTMessageEntryEnvelope business process:

Parameter	Default	Description
Document Tracking	False	When document tracking is enabled for a business process, tracking information is carried with the message throughout the process, and the tracking information is persisted about the message regardless of the persistence level you configured globally for Application.
Set onfault processing	False	Onfault processing allows the process to immediately execute the on-fault activity specified in the process, even if the process has not yet reached that step in the process. For example, if a process fails at step 3, but the on-fault activity is specified in step 7, if onfault processing is enabled, the process proceeds to the step 7 on-fault rather than halting at step 3.
Start mode	async	Asynchronous initiation is selected by default. Starting business processes asynchronously is recommended. Asynchronous mode is standard Application processing, wherein the business process is placed in queue and processed.
Transaction	False	This option instructs Application to treat the entire process as a single transaction so that either all of the steps complete, or, in the event of an error, none of them do. When an error occurs, no data is committed; data returns to its pre-process state. By default, this transaction mode is not enabled.
Queue	4	Application enables you to set performance optimizations by queue, defining queue levels to allocate resources. This number indicates the previously allocated queue level that you want for this business process model for processing.
Persistence Level	System Default	The level of data to retain for generating a status report that describes each step that the business process completes. System default indicates that, for the data, configuration is already defined in Application to retain data.
Recovery Level	Manual	The level of recovery for this business process if the business process should halt during execution. Manual requires you to resume or restart the business process manually.
Document Storage Type	System Default	The level of document storage for messages that process when the business process runs. System Default specifies to store messages in the file system or database, according to how you configured archiving and purging in Application.
Life Span	Life Span Days — 2 Life Span Hours — 0 Life Span Type — System Level Removal Method — Archive	The length of time, in days and hours, to retain the data in Application, along with the life span type and removal method.

Parameter	Default	Description
Complete by Deadline	None Available **Note:** To set a deadline you must change it in the business process.	Complete by – The deadline time, in hours and minutes, by which the business process must complete process once it starts. ◆ First Notification: Hours and Minutes – Whether to receive notification before a business process deadline. ◆ Second Notification: Hours and Minutes – Whether to receive another notification before a business process deadline.
Event Reporting Level	Full	The level of event reporting that is retrieved for this business process when it runs. Full specifies to generate events for the business process, including the business process start and end time, start and end times for all services or services running as a result of this business processes, and any resulting errors and exceptions.

The following BPML code makes up the SWIFTMessageEntryEnvelope business process:

```
<process name="SWIFTMessageEntryEnvelope">
  <sequence>
    <operation name="EDI Encoder">
      <participant name="EDIEncoder"/>
      <output message="EDIEncoderTypeInputMessage">
        <assign to="AccepterLookupAlias" from
="//ProcessData/AccepterLookupAlias/text()" />
        <assign to="EDIStandard">SWIFT</assign>
        <assign to="ReceiverID" from ="//ProcessData/ReceiverId/text()" />
        <assign to="SenderID" from ="//ProcessData/SenderId/text()"/>
        <assign to="MODE">IMMEDIATE</assign
        <assign to="." from="*"></assign>
      </output>
      <input message="inmsg">
        <assign to="." from="*"></assign>
      </input>
    </operation>
    <operation name="EDI Envelope">
      <participant name="EDIEnvelope"/>
      <output message="EDIEnvelopeTypeInputMessage">
        <assign to="MODE">IMMEDIATE</assign>
        <assign to="." from="*"></assign>
      </output>
      <input message="inmsg">
        <assign to="." from="*"></assign>
      </input>
    </operation>
    <operation name="Invoke Business Process Service">
      <participant name="InvokeBusinessProcessService"/>
      <output message="Xout">
        <assign to="INVOKE_MODE">SYNC</assign>
        <assign to="." from="*"></assign>
      </output>
      <input message="Xin">
```

```
      <assign to="." from="*"></assign>
    </input>
  </operation>
  </sequence>
</process>
```

Before Using the SWIFTMessageEntryEnvelope Business Process

Before you use the SWIFTMessageEntryEnvelope business process, you must enable Document Tracking in the Business Process Manager for the SWIFTMessageEntryEnvelope business process.

Creating SWIFTNet Maps

Overview

The Map Editor enables you to map SWIFTNet documents.The Map Editor generates a file layout for you using the components and message types that you select. The Map Editor-generated SWIFT map consists of groups, records, composites, and fields that are comparable to parameters that are defined by SWIFT. See *How SWIFTNet Terminology Correlates with the Map Editor* on page 102.

Note: You need to download the Map Editor component from Application to use it.

Map Editor allows you to modify the map components by using the Deactivate, Promote, Split, Copy, Cut, and Paste functions.

You can create a map for all SWIFTNet Standards Release messages loaded into the standards database through the Map Editor. All messages are validated by Application for syntax (for example, field types, field lengths, and so forth). See *Overview* on page 6 for information on which messages are validated by Application for syntax *and* semantics. If you create a map for a message that is not currently validated by Application, when the Map Editor is capable of updating your existing map with message validation rules in a future release, any extended rules you have created will exist outside the validation block (and thus will always run regardless of whether validation is enabled).

When you create a map, for both the input and output sides, you choose whether you want to use a preloaded standard, an existing file format, or wants to create a new file format for that side of the map (including selecting from standards that you have previously downloaded to the standards database. The preloaded standards are automatically loaded when you download the Map Editor to your machine and include SWIFTNet FUNDS, SWIFTNet Trade Services, SWIFTNet Exceptions and Investigations, and SWIFTNet Cash Reporting.

However, if you want to use a specialized version of a message that is not available in the Application SWIFTNet data dictionary, it may be appropriate for you to define the SWIFTNet map yourself. Alternatively, you could use an MT or MX that is similar to what is required and customize it yourself.

Application also performs SWIFTNet MX validations. Validations are defined in the **translator_swift_mx.properties** property file and generated maps define the parameters used to look up validates in this property file. SWIFT MT validations are also applied to SWIFT MX messages for currencies, country codes, BIC/BEI addresses, and International Bank Account Number checksum validation (IBAN). You can define code lists for currencies, countries, and BIC or BEI addresses (that are validated against the SWIFT_Addresses code list). IBAN data contains a country code that is validated against the SWIFT_Countries code list in Application, and additional IBAN validation is handled internally by the translator.

Application supplies you with the following statements to be used with SWIFTNet maps and any other data format:

✦ cerror

✦ occurrencetotal

✦ resetoccurrencetotal

✦ sum

✦ sumtotal

See *Alphabetic Language Reference* for all the available extended rules.

To ensure that extended rules for SWIFTNet maps are only run when validation is enabled, place your custom extended rule code in a validation block. See *Creating Extended Rules for SWIFTNet Maps* on page 112.

How SWIFTNet Terminology Correlates with the Map Editor

The terminology used by SWIFTNet differs from that used in the Map Editor. This table lists the SWIFTNet terms and how they correspond to map components in the Map Editor:

SWIFTNet Terminology	Corresponding Map Editor Terminology
MT or MX (File)	SWIFT file format (the top map component on the input and output sides of a map)
Sequence or an implicit group of repeating field tags	Group
Field tag	SWIFT Record
Subfield that is an "OR" option or a group of related subfields that occur in a sequence, and they are also groups of related subfields that occur in a sequence (for example, each SWIFTBIC address is defined as a "composite" consisting of a number of subfields such as branch code, location, and so forth)	Composite
Subfield	Field
Component or a group of SWIFT components that define a SWIFT subfield	Field

SWIFTNet Components in the Map Editor

The following table lists the components that make up the SWIFTNet layout in the Map Editor, the icons that represent the components, and descriptions of the components. For information about adding a map component to a layout, see *Map Editor Basics*.

Component	Icon	Description
SWIFT root element		The *SWIFT root element* represents the MT or MX that Application is mapping. At the SWIFT file root element, you define the message type and encoding. It is a group and can contain groups and SWIFT records.
Group		A *group* is a looping structure that contains a sequence or an implicit group of repeating field tags (in Map Editor a group is related records and groups that repeat in sequence until either the group data ends, or the maximum number of times that the loop is permitted to repeat is exhausted).
		A group that is subordinate to another group is a subgroup (and corresponds to a nested looping structure, a loop within a loop).
		When a group contains an extended rule or a standard rule, an asterisk appears to the right of the group icon.
SWIFT Record		A *SWIFT record* contains a field tag (in Map Editor a SWIFT record is a group of related fields or composite data elements that combine to communicate useful data). A SWIFT record can occur once or can repeat multiple times.
		Note: If a SWIFT record occurs more than once in a map, it is identified by its name <ID>. The second and subsequent occurrences are identified by <ID>:*n*, where *n* is the number of the occurrence in the map.
Composite		A *composite* is a subfield that is an "OR" option or a group of related subfields that occur in a sequence (in Map Editor a composite is a data element that contains two or more component data elements or subelements). They are also groups of related subfields that occur in a sequence. For example, each SWIFTBIC address is defined as a "composite" consisting of a number of subfields (e.g. branch code, location).
		A composite can occur once or repeat multiple times.
		For example, each SWIFTBIC address is defined in the Map Editor as a composite that consists of a number of fields (SWIFT subfields) such as branch code, location, and so forth.
		Note: If a composite occurs more than once in a map, it is identified by its name <ID>. The second and subsequent occurrences are identified by <ID>:*n*, where *n* is the number of the occurrence in the map.
		A *repeating composite* is a related group of fields that have the ability to loop as a whole (occur more than once) within a particular SWIFT record. To enable a composite to repeat multiple times within a SWIFT record, each occurrence of the composite must be separated by a start and end delimiter.
		Note: SWIFTNet does not use the repeating option, although it is available to you.

Component	Icon	Description
Field		A *field* is a subfield or a group of SWIFT components (the smallest piece of information defined by the SWIFTNet standard) that define a SWIFT subfield. A field can have different meanings depending on the context. In other data formats in the Map Editor, a field is not considered to have useful meaning except in the larger context of the record that contains it. However, fields used in the Map Editor to represent SWIFTNet subfields and components contain useful and discrete information.
		Note: If a field occurs more than once in a map it is identified by its name <ID>. The second and subsequent occurrences are identified by <ID>:*n*, where *n* is the number of the occurrence in the map.
		A *repeating field* is a field with the ability to loop (occur more than once) within a particular SWIFT record. To enable a single field to repeat multiple times within a SWIFT record, each occurrence of the field must be separated by a start and end delimiter. The use of start and end delimiters help the translator determine where subfields and components are defined within a field tag.
		When a field has a link performed against it, a red check mark appears over the field icon.
		When a field contains an extended rule or a standard rule, an asterisk appears to the right of the field icon.

Downloading and Installing the SWIFT Standards Database

Before you install the SWIFT standards database on your desktop, consider these guidelines:

✦ Download the Map Editor.

✦ For the most current version of the SWIFT standard, contact Sterling Commerce Customer Support.

✦ Be sure your desktop meets the Windows Client requirements listed in the Application *System Requirements*.

To download and install the standards database:

1. From the Application **Deployment** menu, select **Standards**.

2. In the Download and Install section next to Download SWIFT Standards, click **Go!**

3. In the **File Download** dialog box, select a download option, then click **OK**.

   ◆ If you choose to run the file click **Run** and the operating system downloads the files immediately.

   ◆ If you choose to save the file, the operating system prompts you to save the file. Browse to the location where you want to download the file and click **OK**. If you want to continue installing, run the file you just saved from the location you specified.

4. In the Security Warning page, select **Always trust content from Sterling Commerce (Mid America), Inc.** if you do not want to see similar security messages in the future when you download software from Sterling Commerce. Click **Yes**.

5. In the Welcome window, click **Next**.

6. In the Choose Destination Location window, select where you want to install the standards database:

   ◆ If you accept the default location, click **Next**.

◆ If you want to specify a different location, click **Browse**, specify the path to the folder, click **OK**, and click **Next**.

If you specify a folder name that does not exist, Application displays a message asking if you want to create that folder.

7. In the Select Components window, verify that **SWIFTStandardDatabase** is selected and click **Next**.

The download wizard installs the standards database (SWIFT_2006.mdb).

8. In the Setup Complete window, click **Finish**.

Creating a File Layout from an MT or Market Practice

When you create a new map, you typically use the map wizard that creates a layout for you based on an MT or Market Practice from the standards database. The wizard saves you the time and effort to create the SWIFT side of the map yourself, and minimizes the risk of having an invalid standard format for a message.

When you create a map, for both the input and output sides, you choose whether you want to use a preloaded standard, an existing file format, or wants to create a new file format for that side of the map (including selecting from standards that you have previously downloaded to the standards database.

Note: You can create a map for all SWIFTNet messages through the Map Editor, and all messages are validated by Application for syntax (that is, field types, field lengths, and so forth).

To create a file layout from an MT, MX, or Market Practice:

1. From the Map Editor **File** menu, select **New**.

2. In the **New Map Wizard**, complete the questions in the first window and click **Next**.

Note: Be sure that **Sterling Integrator** is selected in the **What type of map are you creating** list.

3. If you are translating from SWIFTNet, in the Input Format window, Create a new data format using this standards, select **SWIFT (Society for Worldwide Interbank Financial Telecommunications)** and click **Messages**.

Note: The **Create a new data format using this** selection allows you to use a preloaded standard (that is, you do not have to download the standard to the Application database) or use the standards that were downloaded through the standards database.

4. Click **Next**.

5. Select the type of message to auto-generate:

◆ Standard SWIFT Message Type

◆ Market Practice or Fund Template

◆ Custom Market Practice or Fund Template

Note: If you select Custom Market Practice or Fund Template, you must have already created a custom Market Practice or Fund Template to have it listed as an available option.

6. If you are translating an MT, select the **GIS SWIFT Standard** ODBC data source (which contains the SWIFTNet standards database) and click **Next**.

7. If you are translating a Market Practice, select the **GIS SWIFTMP Customer** ODBC data source (which contains the SWIFTNet preloaded market practices) and click **Next**.

8. If you are translating from an MT, select the MT that you want to use and select the version, and click **Next**.

9. If you are translating from a preloaded Market Practice or fund template, select the MT you want to modify and click **Next**.

10. If you are translating from SWIFTNet, click **Finish** to load the MT you selected.

11. Click **Next**.

12. If you are translating to SWIFTNet, in the Output Format window, select **SWIFT (Society for Worldwide Interbank Financial Telecommunications)** and click **Customize**. If you are translating from another format, select that format and continue to the next screen.

13. Click **Next**.

14. If you are translating to SWIFTNet, select the **GIS SWIFT Standard** ODBC data source (which contains the SWIFTNet standards database) and click **Next**.

15. If you are translating to SWIFTNet, select the MT that you want to use and select the version, and click **Next**.

16. If you are translating to SWIFTNet, click **Finish** to load the MT you selected.

17. Click **Next**.

18. If you are translating to SWIFTNet, click **Finish**. The Map Editor displays the new map in the Map Editor window.

19. Use Autolink to link your fields prior to using the Link function or creating standard or extended rules.

20. Use the Rule Library Manager to add a reusable extended rule library that can then be added to any map so you do not have to redefine extended rules for each map you create.

Note: The SWIFT extended rules libraries that are installed with Application, contain all the extended rules necessary to carry out the business logic for SWIFT messages.

Creating a Custom Market Practice or Fund Template

You have the capability to create a custom market practice or fund template by using the Map Editor to create a message template of the same message type you want, modifying it accordingly, and using the SWIFT Rule Importer to import the map into Application.

To create a custom Market Practice or Fund Template:

1. From the Map Editor **File** menu, select **New**.

2. In the **New Map Wizard**, complete the questions in the first window and click **Next**.

Note: Be sure that **Sterling Integrator** is selected in the **What type of map are you creating** list.

3. In the Input Format window, Create a new data format using this standards, select **SWIFT (Society for Worldwide Interbank Financial Telecommunications)** and click **Messages**.

4. Click **Next**.

5. Select **Standard SWIFT Message Type** as the type of message to auto-generate.

6. Select the **GIS SWIFT Standard** ODBC data source (which contains the SWIFTNet standards database) and click **Next**.

7. Select the MT that you want to use and select the version, and click **Next**.

8. If you are translating from SWIFTNet and have selected a common group MT, select the MT for copy fields and click **Next**.

9. If you are translating from SWIFTNet, click **Finish** to load the MT you selected.

10. Click **Next**.

11. In the Output Format window, select **SWIFT (Society for Worldwide Interbank Financial Telecommunications)** and click **Customize**. If you are translating from another format, select that format and continue to the next screen.

12. Click **Next**.

13. Select the **GIS SWIFT Standard** ODBC data source (which contains the SWIFTNet standards database) and click **Next**.

14. Select the *same MT that you selected for the Input Format* and select the version, and click **Next**.

Note: By selecting the same MT on both the input and output sides of the map, you are creating a "passthrough" map.

15. If you are translating to SWIFTNet and have selected a common group MT, select the MT for copy fields and click **Next**.

16. Click **Finish** to load the MT you selected.

17. Click **Next**.

18. Click **Finish**. The Map Editor displays the new map in the Map Editor window.

19. On the Output side of the map, right-click the SWIFT Properties icon and select **Properties**.

20. Select the **Message Type** tab.

21. In **Market Practice ID**, type the unique identifier for the market practice. This identifier distinguishes the map from the standard SWIFT message type.

Note: This identifier must be 6 upper-case alphanumerics with no spaces. You create the unique Market Practice ID.

22. Use the **Save As** function to save the modified map under a new name.

23. Open a command window and change to the directory where you installed the SWIFT Standards Database.

24. Then, from the Map Editor **File** menu, select **New** so you can use the customer Market Practice or Fund Template to create a new map.

25. In the **New Map Wizard**, complete the questions in the first window and click **Next**.

Note: Be sure that **Sterling Integrator** is selected in the **What type of map are you creating** list.

26. In the Input Format window, Create a new data format using this standards, select **SWIFT (Society for Worldwide Interbank Financial Telecommunications)** and click **Messages**.

27. Click **Next**.

28. Select **Custom Market Practice or Fund Template** as the type of message to auto-generate.

29. Select the **GIS SWIFT Standard** ODBC data source (which contains the SWIFTNet standards database) and click **Next**.

30. Select the custom Market Practice or Fund Template and click **Next**.

31. Follow the prompts to complete the map selection.

32. Click **Finish**. The Map Editor displays the new map in the Map Editor window.

33. Use Autolink to link your fields prior to using the Link function or creating standard or extended rules.

34. Modify the map to implement the additional restrictions related to the desired Market Practice or Fund Template.

35. Use the Rule Library Manager to add a reusable extended rule library that can then be added to any map so you do not have to redefine extended rules for each map you create.

Note: The SWIFT extended rule libraries that are installed with Application contain all the extended rules necessary to carry out the business logic for SWIFT messages.

36. For the SWIFT Rule Importer command script, set the Java_home variable.

37. Run the SWIFT Rule Importer from the command prompt using the following syntax:

    `SWIFTRuleImporter MapName.mxl`

38. Save your map using the following naming convention:

`SWIFT_<version>_<messageType>[_<messageExtension>]_<marketPracticeID>.mxl`

Note: In this syntax, version is synonymous with year. For Market Practices, you do not need a messageExtension since all market practice maps are in the 5xx series. The map must be saved in MXL format.

39. Compile the map.

40. If necessary, update your property files with code word and qualifier validations that are specific to the Market Practice.

Note: The properties file to update is **translator_swift_mp_customer_year.properties**, located in the **<install_dir>/properties** directory.

Creating a SWIFTSolutions (MX) Map using SWIFTNet FUNDS, SWIFTNet Trade Services, SWIFTNet Exceptions and Investigations, and SWIFTNet Cash Reporting

When you create a SWIFTSolutions (MX) map, for both the input and output sides, you choose whether you want to use a preloaded standard, an existing file format, or wants to create a new file format for that side of the map (including selecting from standards that you have previously downloaded to the standards database. The preloaded standards are downloaded when you download the Map Editor to your machine and include SWIFTNet FUNDS, SWIFTnet Trade Services, SWIFTNet Exceptions and Investigations, and SWIFTNet Cash Reporting.

For a complete list of the preloaded standards consult the Map Editor New Map Wizard.

Note: If the map you are creating contains greater than 20,000 objects, you will receive a message noting that this map contains a very large number of objects. For best performance, it is recommended that

you consider whether any unnecessary objects in the map can be removed, do not expand the entire object tree—expand only the section of the tree you are currently mapping, consider using the "Show links to or from the currently selected element" option instead of the "Show links to or from all visible elements" option, and save the map using the.MAP file format (using the Save As function).

To create a map using a preloaded SWIFTNet Fund template:

1. From the Map Editor **File** menu, select **New**.

2. In the **New Map Wizard**, complete the questions on the first screen.

Note: To use XML schemas, the map type must be Sterling Integrator.

3. If you are translating from a SWIFTNet Fund template, from the **Create a new format using this standard** list, select **SWIFTNet Funds** (or SWIFTnet Trade Services, SWIFTNet Exceptions and Investigations, or SWIFTNet Cash Reporting) and click **Messages**. If you are translating from another format, select that format and continue to the next screen.

4. Complete the Map Wizard by selecting the version of the standard, the desired message, the maximum length of the data elements, and whether to build codelists for enumerated attributes, and click **Next**.

Note: The default for the **Build code lists for enumerations** check box is cleared because using code lists may be redundant and thus your map is smaller and more efficient without them.

5. Click **Next**.

6. If you chose to import an XML schema, do not modify any of the subelements selected in the display. *This list is included for your information only.* Click **Next**.

   If you are using an XML schema and the XML parser detects any errors, the messages are displayed in an error window.

7. Click **Finish**.

8. If you are translating to a SWIFTNet Fund template, on the Output screen (from the **Create a new format using this standard** list), select select **SWIFTNet Funds** (or SWIFTnet Trade Services, SWIFTNet Exceptions and Investigations, or SWIFTNet Cash Reporting) and click **Messages**. If you are translating from another format, select that format and continue to the next screen.

9. Complete the Map Wizard by selecting the version of the standard, the desired message, the maximum length of the data elements, and whether to build codelists for enumerated attributes, and click **Next**.

Note: The default for the **Build code lists for enumerations** check box is cleared because using code lists may be redundant and thus your map is smaller and more efficient without them.

10. Click **Next**.

11. If you chose to import an XML schema, do not modify any of the subelements selected in the display. *This list is included for your information only.* Click **Next**.

    If you are using an XML schema and the XML parser detects any errors, the messages are displayed in an error window.

12. Click **Finish**.

    The **XML Load Warnings** dialog box opens if there are any errors. If the Map Editor made changes to the DTD to make it compliant with Application, it indicates the changes. Click **OK**.

13. Continue with the **New Map Wizard** as directed. When you click **Finish**, the Map Editor displays the new map in the Map Editor window.

SWIFTNet Syntax Validation

SWIFT field tag syntax validation is defined in the Map Editor and stored stored in the translator_swift_2007.properties file. This properties file contains the validation syntax for the field tags, as defined by the SWIFT standards (for example, field 73 is 35x['CRLF'35x]0-5), and may not be edited. The validation syntax for any given record is displayed in the Field Syntax parameter on the SWIFT Record Properties dialog box (SWIFT Validation tab). This parameter is automatically populated when you create a map using the SWIFTNet standards data dictionary. The convention of the syntax conforms exactly to the SWIFT documentation.

SWIFT also uses special validation functions such as <CUR>, <SWIFTBIC>, and <NON-SWIFTBIC>, which are components of a field tag. Each of these special functions has an expanded syntax that is validated by the Application translator. When these special validation functions are used, the special function syntax is displayed in the **Field Syntax** text box on the **SWIFT Record Properties** dialog box. The currency (<CUR>) and sender/receiver addresses (<SWIFTBIC>, <NON-SWIFTBIC>) must also be validated against legal code word lists. These code lists are defined in the SWIFTNet standards data dictionary and used by Application.

Note: You will need to maintain these code lists in Application (not in the Map Editor). See *Creating and Using Code Lists* for more information.

Additionally, the Map Editor and the translator allows complex syntax validation, which can be used for any data format. The Map Editor provides the Free Format option on the Field Properties Validation tab to enable you to specify that any characters are acceptable in the field, and the translator does not check the characters for compliance. To define a complex syntax, you just need to use the Free Format option for a String data type and define the syntax.

To define a complex syntax when you create a SWIFT map without using the SWIFTNet standards data dictionary or want to edit the syntax for a record:

1. In the Map Editor, right-click the record for which you want to define or edit the syntax.

2. In the **SWIFT Record Properties** dialog box, click the **SWIFT Validation** tab.

3. In **Field Syntax**, type the necessary complex syntax such as **3a[2!n]**, which indicates that the String should consist of zero to three uppercase letters followed by an optional two digits.

4. Click **OK** to save the information.

SWIFT MX Validation

Application performs SWIFTNet MX validations. Validations are defined in the **translator_swift_mx.properties** property file and generated maps define the parameters used to lookup validates in this property file.

The following is an example entry from the **translator_swift_mx.properties** property file for Currency validation. In this example, currency validation is performed on the XML element **Amt**, attribute **Ccy**.

```
swiftMX.tsmt.009.001.02.Amt.Ccy=CUR
```

The following is an example entry from the **translator_swift_mx.properties** property file for Country validation. In this example, country validation is performed on the XML element **PdctOrgn**.

```
swiftMX.tsmt.009.001.02.PdctOrgn=CC
```

The following is an example entry from the **translator_swift_mx.properties** property file for IBAN validation. In this example, IBAN validation is performed on the XML element **IBAN**.

```
swiftMX.tsmt.009.001.02.IBAN=IBAN
```

The following is an example entry from the **translator_swift_mx.properties** property file for BIC validation. In this example, BIC validation is performed on the XML element **BIC**.

```
swiftMX.tsmt.009.001.02.BIC=BIC
```

When you create a SWIFT MX map, the parameters used to look up validations in the property file are defined on the Map Details dialog box. This table describes the Map Details parameters and their function in MX validation:

Map Details Parameter	Description of MX Validation Function
Agency (description field)	**SWIFT-MX** identifies the map as requiring SWIFT MX validation.
Version	The version of the SWIFTNet solution (for example, 1-0).
Transaction (description field)	The schema name is used as a unique key to lookup validation entries. For example, 1-0.tsmt.009.001.02 is a unique key for a Trade Services Baseline Amendment Request in which: ◆ 1-0 is the version of the SWIFTNet solution (1.0) ◆ tsmt is the SWIFTNet solution (for example, Trading Services) ◆ 009.001.02 is the schema name

Error Codes for MX Validation

The following error codes are logged when MX validation errors occur. These errors are logged in the translator report and in the SWIFT Message Editor when you are using the Document Repair function:

Error Code	Description
Sw.Stds.D00001	Invalid BIC code
Sw.Stds.D00002	Invalid BEI code
Sw.Stds.D00003	Invalid IBAN code
Sw.Stds.D00004	Invalid country code
Sw.Stds.D00005	Invalid active currency code
Sw.Stds.D00006	Invalid active or historic currency code
Sw.Stds.D00007	Invalid currency code or too many decimal digits
Sw.Stds.D00008	Invalid BIC or BEI code

Creating Extended Rules for SWIFTNet Maps

In the next release of SWIFTNet there will be support to update existing maps to include or add any message rule validation changes. This future update feature will impact you only if you want custom code in the extended rules (in existing maps) included in the validation block that only runs when validation is enabled. The update feature places custom extended rule code outside of the validation block. Thus, if you want to create custom rules used only during validation, then you need to follow the example below when you write extended rules for SWIFTNet maps to ensure they are only run when validation is enabled:

```
if validation = 1 then
begin
    {type your custom validation extended rule code}
end
```

Extended Rules Used with SWIFTNet Maps

The following new extended rules are used with SWIFTNet maps, though they can also be used with any other data format:

✦ cerror

✦ occurrencetotal

✦ resetoccurrencetotal

✦ sum

✦ sumtotal

See *Alphabetic Language Reference* for information on all the available extended rules.

Using Autolink and Link

The Autolink function automatically creates links between input and output fields that have the same name or which contain logically equivalent business data. This functionality can be used regardless of which format you have selected for the input and output sides of your map. See *Using Autolink with the Map Editor* for more information on using this function. You can choose to link by either Field Name or Business Name.

Note: To increase the likelihood that the links in your maps are valid, in the Preferences dialog box Confirmations tab, select the link objects at different levels and link objects with different maximum usages confirmations. See *Using Autolink with the Map Editor* for more information on using this function.

Just like with the Link function, the link between two map components is represented visually with a connecting line. See *Creating Simple Links* for more information on the Link function.

Note: An erroneous message is displayed when you deactivate a group that contains linked fields in the Map Editor. The message displayed is: "The object is part of one or more links. Deleting the object will remove the links. Do you want to continue?" This message should read "The object is part of one or more links. Deactivating the object will remove the links..." If you receive this message when

deactivating a linked map component, be assured that the group object is not deleted, and you should ignore the erroneous message.

Using Extended Rule Libraries with SWIFTNet

This section describes how to use the extended rule library and the properties of the dialog boxes that comprise its functionality. A rules library (used with SWIFTNet and any other data format) contains a list of rules in a separate file outside of the Map Editor source. Map Editor stores the name of the library in its source file, so when you open a map the library is also loaded. When you compile a map, the library rules that are referenced in the map are also compiled. This enables you to create a library of extended rules and then add it to any other map, so you do not have to recreate those extended rules after the first time. You can use this functionality with any data format.

Note: The SWIFT extended rules libraries are automatically installed with Application (and checked in), contains all the extended rules necessary to carry out the business logic for SWIFT messages.

Semantic validation rules for Market Practices are stored in an extended rule library separate from the library that implements the SWIFT standard semantic validation rule

Only the extended rule libraries referenced by a map are compiled into the TXO translation object.

This functionality minimizes the impact to users when, for example, SWIFT updates their messages—without the rule library you would need to update the extended rules for each updated map (correlating to the updated messages), but using the extended rule library you just update the library and then use the library with all the applicable maps.

When you view the checked in libraries through the Extended Rule Library check in interface, you are also able to obtain a list of all the maps that use each library.

The extended rules library can contain many rules. An extended rule consists of a declarations section followed by a statements section. The *declarations* section is required only if you use additional variables. The declarations section is where you declare the names and types of any variables you use either in the extended rule. The *statements* section is where you define the actions that you want the extended rule to run.

When calling a rule library function, you can pass parameters.

You must declare any variables that are not already defined as part of the input or output specification of the map before you use those variables in an extended rule. For the extended rule libraries, you typically use global variables that are passed as parameters.

Rule libraries are versioned resources. When you create a new rule library you need to check it in to Application just like you need to check in maps. This also enables you to check out, version, and delete extended rule libraries. Furthermore, when you view the checked in libraries through the Extended Rule Library check in interface, you can also see all the maps that use each library. This is very important because it enables you to easily view a list of the maps that will need to be recompiled if you change an extended rule in a library (you would recompile all the maps that use that particular library).

Additionally, you can import and export extended rule libraries into Application using the Resource Manager.

You can call an extended rule from a library in any extended rule in a map.

See *Extended Rule Libraries* for more information on this functionality.

The extended rule libraries that are preloaded for use with SWIFTNet are as follows:

Library	ERL File Name	Description
SWIFT_2006	SWIFT_2006.erl	Contains rules to validate SWIFT 2006 MT and MX messages.
SWIFT_2007	SWIFT_2007.erl	Contains rules to validate SWIFT 2007 MT and MX messages.
SWIFTMX	SWIFTMX.erl (Common Rule Library)	Contains rules to validate MX messages.
SWIFTMX_FUNDS	SWIFTMX_Funds_v4-0.erl	Contains rules to validate SWIFT Funds 4.0 messages.
SWIFTMX_CashReporting	SWIFTMX_CashReporting_3-1.erl	Contains rules to validate SWIFT Cash Reporting 3.1 messages.
SWIFTMX_EI	SWIFTMX_EI.erl	Contains rules to validate SWIFT Exceptions and Investigations messages.
SWIFTMX_TradeServices	SWIFTMX_TradeServices.erl	Contains rules to validate SWIFT Trade Services messages.

Calling a Rule from an Extended Rule Library in a Map

You can call a rule from any extended rule library (that is currently checked in to Application) in any extended rule in your map. The syntax you use to call a rule from a library is:

```
call library_name.rule_name(parameter1, parameter2, parameter3)
```

In this syntax, **library_name** is the name of the extended rule library. For example, if the library is SWIFT_2006.erl, the library_name is **SWIFT**. The **rule_name** is the name of the rule that you defined in the Library Rule dialog box. There is no limit on the number of parameters you can use.

Note: You can have multiple rule libraries with the same name and different version numbers, but you can only use one rule library of the same name in a map (the last version of that rule library that was checked in to the system).

The syntax you use to call a rule with a return value set is:

```
integer i;
i = call library_name.rule_name parameter1
```

In this syntax, **i** is the return value set.

Truncating Number Fields When Converting Strings to Numbers

By default, the translator trims all trailing zeros from output values. For example, 80000.0100 will be output as 8000.01. This behavior is not desirable for financial standards such as SWIFTNet, for which trailing zeros represent a specific amount for a particular currency.

Therefore, the **storage.keepTrailingZeros** property in the **translator.properties** file enables you to specify whether or not trailing zeros are kept on any numeric field on the output side of the map (numeric fields that have been converted to strings).

By default trailing zeros are trimmed (storage.keepTrailingZeros=false). If you set this property to True, this will preserve trailing zeroes. Therefore, by default the value 3.142000 is truncated to 3.142, but if you set the **storage.keepTrailingZeros** property to True, the resulting string value is 3.142000.

To set the **storage.keepTrailingZeros** property to True:

1. In the *install_dir*/properties directory, locate (or create, if necessary) the **customer_overrides.properties** file.

2. Open the **customer_overrides.properties** file in a text editor.

3. Add the property that you want to override, using the following format:

   ```
   translator.storage.keepTrailingZeros=True
   ```

4. Save and close the **customer_overrides.properties** file.

5. Restart Application to use the new values.

Using BigDecimal

Application allows you to use either Java double primitive or BigDecimal data types for real numbers. BigDecimal can be used regardless of the standard you are using, but if you are using a financial standard like SWIFT we highly recommend that you use BigDecimal. See *BigDecimal Support for Real Numbers* for more information on using this functionality. When processing SWIFT messages with BigDecimal mode turned off, real values (such as amounts) may be rounded when those messages are processed through the SWIFT Message Editor. When rounding occurs, the changed values are not reported.

If you are using BigDecimal mode on a per map basis for the SWIFT standard, you also need to enable BigDecimal (turn it on) for any validation pass-through maps you are using. It is also possible to use double primitive mode by default, but to enable BigDecimal mode on a per-map basis. Validation pass-through maps should have BigDecimal mode enabled (turned on).

The validation pass-through maps used with Application are as follows:

MT Maps

✦ SWIFT_2002_100.mxl

✦ SWIFT_2005_097.mxl

✦ SWIFT_2005_101.mxl

✦ SWIFT_2005_102.mxl

✦ SWIFT_2005_102_STP.mxl

✦ SWIFT_2005_103.mxl

✦ SWIFT_2005_103_STP.mxl

✦ SWIFT_2005_104.mxl

✦ SWIFT_2005_105.mxl

✦ SWIFT_2005_106.mxl

- ✦ SWIFT_2005_107.mxl
- ✦ SWIFT_2005_110.mxl
- ✦ SWIFT_2005_111.mxl
- ✦ SWIFT_2005_112.mxl
- ✦ SWIFT_2005_190.mxl
- ✦ SWIFT_2005_191.mxl
- ✦ SWIFT_2005_192.mxl
- ✦ SWIFT_2005_195.mxl
- ✦ SWIFT_2005_196.mxl
- ✦ SWIFT_2005_198.mxl
- ✦ SWIFT_2005_199.mxl
- ✦ SWIFT_2005_200.mxl
- ✦ SWIFT_2005_201.mxl
- ✦ SWIFT_2005_202.mxl
- ✦ SWIFT_2005_203.mxl
- ✦ SWIFT_2005_204.mxl
- ✦ SWIFT_2005_205.mxl
- ✦ SWIFT_2005_206.mxl
- ✦ SWIFT_2005_207.mxl
- ✦ SWIFT_2005_210.mxl
- ✦ SWIFT_2005_256.mxl
- ✦ SWIFT_2005_290.mxl
- ✦ SWIFT_2005_291.mxl
- ✦ SWIFT_2005_292.mxl
- ✦ SWIFT_2005_295.mxl
- ✦ SWIFT_2005_296.mxl
- ✦ SWIFT_2005_298.mxl
- ✦ SWIFT_2005_299.mxl
- ✦ SWIFT_2005_300.mxl
- ✦ SWIFT_2005_303.mxl
- ✦ SWIFT_2005_304.mxl
- ✦ SWIFT_2005_305.mxl
- ✦ SWIFT_2005_306.mxl
- ✦ SWIFT_2005_307.mxl
- ✦ SWIFT_2005_308.mxl
- ✦ SWIFT_2005_320.mxl

- ✦ SWIFT_2005_321.mxl
- ✦ SWIFT_2005_330.mxl
- ✦ SWIFT_2005_340.mxl
- ✦ SWIFT_2005_341.mxl
- ✦ SWIFT_2005_350.mxl
- ✦ SWIFT_2005_360.mxl
- ✦ SWIFT_2005_361.mxl
- ✦ SWIFT_2005_362.mxl
- ✦ SWIFT_2005_364.mxl
- ✦ SWIFT_2005_365.mxl
- ✦ SWIFT_2005_380.mxl
- ✦ SWIFT_2005_381.mxl
- ✦ SWIFT_2005_390.mxl
- ✦ SWIFT_2005_391.mxl
- ✦ SWIFT_2005_392.mxl
- ✦ SWIFT_2005_395.mxl
- ✦ SWIFT_2005_396.mxl
- ✦ SWIFT_2005_398.mxl
- ✦ SWIFT_2005_399.mxl
- ✦ SWIFT_2005_400.mxl
- ✦ SWIFT_2005_405.mxl
- ✦ SWIFT_2005_410.mxl
- ✦ SWIFT_2005_412.mxl
- ✦ SWIFT_2005_416.mxl
- ✦ SWIFT_2005_420.mxl
- ✦ SWIFT_2005_422.mxl
- ✦ SWIFT_2005_430.mxl
- ✦ SWIFT_2005_450.mxl
- ✦ SWIFT_2005_455.mxl
- ✦ SWIFT_2005_456.mxl
- ✦ SWIFT_2005_490.mxl
- ✦ SWIFT_2005_491.mxl
- ✦ SWIFT_2005_492.mxl
- ✦ SWIFT_2005_495.mxl
- ✦ SWIFT_2005_496.mxl
- ✦ SWIFT_2005_498.mxl

- SWIFT_2005_499.mxl
- SWIFT_2005_500.mxl
- SWIFT_2005_501.mxl
- SWIFT_2005_502.mxl
- SWIFT_2005_503.mxl
- SWIFT_2005_504.mxl
- SWIFT_2005_505.mxl
- SWIFT_2005_506.mxl
- SWIFT_2005_507.mxl
- SWIFT_2005_508.mxl
- SWIFT_2005_509.mxl
- SWIFT_2005_510.mxl
- SWIFT_2005_513.mxl
- SWIFT_2005_514.mxl
- SWIFT_2005_515.mxl
- SWIFT_2005_515_SMPG02.mxl
- SWIFT_2005_516.mxl
- SWIFT_2005_517.mxl
- SWIFT_2005_518.mxl
- SWIFT_2005_519.mxl
- SWIFT_2005_524.mxl
- SWIFT_2005_526.mxl
- SWIFT_2005_527.mxl
- SWIFT_2005_528.mxl
- SWIFT_2005_529.mxl
- SWIFT_2005_535.mxl
- SWIFT_2005_536_IOA001.mxl
- SWIFT_2005_536.mxl
- SWIFT_2005_537.mxl
- SWIFT_2005_538.mxl
- SWIFT_2005_540.mxl
- SWIFT_2005_541.mxl
- SWIFT_2005_542.mxl
- SWIFT_2005_543.mxl
- SWIFT_2005_544.mxl
- SWIFT_2005_545.mxl

- ✦ SWIFT_2005_546.mxl
- ✦ SWIFT_2005_547.mxl
- ✦ SWIFT_2005_548.mxl
- ✦ SWIFT_2005_549.mxl
- ✦ SWIFT_2005_558.mxl
- ✦ SWIFT_2005_559.mxl
- ✦ SWIFT_2005_564.mxl
- ✦ SWIFT_2005_565.mxl
- ✦ SWIFT_2005_566.mxl
- ✦ SWIFT_2005_567.mxl
- ✦ SWIFT_2005_568.mxl
- ✦ SWIFT_2005_569.mxl
- ✦ SWIFT_2005_574_IRSLST.mxl
- ✦ SWIFT_2005_574.mxl
- ✦ SWIFT_2005_574_W8BENO.mxl
- ✦ SWIFT_2005_575.mxl
- ✦ SWIFT_2005_576.mxl
- ✦ SWIFT_2005_577.mxl
- ✦ SWIFT_2005_578.mxl
- ✦ SWIFT_2005_579.mxl
- ✦ SWIFT_2005_581.mxl
- ✦ SWIFT_2005_582.mxl
- ✦ SWIFT_2005_584.mxl
- ✦ SWIFT_2005_586.mxl
- ✦ SWIFT_2005_587.mxl
- ✦ SWIFT_2005_588.mxl
- ✦ SWIFT_2005_589.mxl
- ✦ SWIFT_2005_590.mxl
- ✦ SWIFT_2005_591.mxl
- ✦ SWIFT_2005_592.mxl
- ✦ SWIFT_2005_595.mxl
- ✦ SWIFT_2005_596.mxl
- ✦ SWIFT_2005_598.mxl
- ✦ SWIFT_2005_599.mxl
- ✦ SWIFT_2005_600.mxl
- ✦ SWIFT_2005_601.mxl

◆ SWIFT_2005_604.mxl
◆ SWIFT_2005_605.mxl
◆ SWIFT_2005_606.mxl
◆ SWIFT_2005_607.mxl
◆ SWIFT_2005_608.mxl
◆ SWIFT_2005_609.mxl
◆ SWIFT_2005_643.mxl
◆ SWIFT_2005_644.mxl
◆ SWIFT_2005_645.mxl
◆ SWIFT_2005_646.mxl
◆ SWIFT_2005_649.mxl
◆ SWIFT_2005_690.mxl
◆ SWIFT_2005_691.mxl
◆ SWIFT_2005_692.mxl
◆ SWIFT_2005_695.mxl
◆ SWIFT_2005_696.mxl
◆ SWIFT_2005_698.mxl
◆ SWIFT_2005_699.mxl
◆ SWIFT_2005_700.mxl
◆ SWIFT_2005_701.mxl
◆ SWIFT_2005_705.mxl
◆ SWIFT_2005_707.mxl
◆ SWIFT_2005_710.mxl
◆ SWIFT_2005_711.mxl
◆ SWIFT_2005_720.mxl
◆ SWIFT_2005_721.mxl
◆ SWIFT_2005_730.mxl
◆ SWIFT_2005_732.mxl
◆ SWIFT_2005_734.mxl
◆ SWIFT_2005_740.mxl
◆ SWIFT_2005_742.mxl
◆ SWIFT_2005_747.mxl
◆ SWIFT_2005_750.mxl
◆ SWIFT_2005_752.mxl
◆ SWIFT_2005_754.mxl
◆ SWIFT_2005_756.mxl

- ✦ SWIFT_2005_760.mxl
- ✦ SWIFT_2005_767.mxl
- ✦ SWIFT_2005_768.mxl
- ✦ SWIFT_2005_769.mxl
- ✦ SWIFT_2005_790.mxl
- ✦ SWIFT_2005_791.mxl
- ✦ SWIFT_2005_792.mxl
- ✦ SWIFT_2005_795.mxl
- ✦ SWIFT_2005_796.mxl
- ✦ SWIFT_2005_798.mxl
- ✦ SWIFT_2005_799.mxl
- ✦ SWIFT_2005_800.mxl
- ✦ SWIFT_2005_801.mxl
- ✦ SWIFT_2005_802.mxl
- ✦ SWIFT_2005_810.mxl
- ✦ SWIFT_2005_812.mxl
- ✦ SWIFT_2005_813.mxl
- ✦ SWIFT_2005_820.mxl
- ✦ SWIFT_2005_821.mxl
- ✦ SWIFT_2005_822.mxl
- ✦ SWIFT_2005_823.mxl
- ✦ SWIFT_2005_824.mxl
- ✦ SWIFT_2005_890.mxl
- ✦ SWIFT_2005_891.mxl
- ✦ SWIFT_2005_892.mxl
- ✦ SWIFT_2005_895.mxl
- ✦ SWIFT_2005_896.mxl
- ✦ SWIFT_2005_898.mxl
- ✦ SWIFT_2005_899.mxl
- ✦ SWIFT_2005_900.mxl
- ✦ SWIFT_2005_910.mxl
- ✦ SWIFT_2005_920.mxl
- ✦ SWIFT_2005_935.mxl
- ✦ SWIFT_2005_940.mxl
- ✦ SWIFT_2005_941.mxl
- ✦ SWIFT_2005_942.mxl

- SWIFT_2005_950.mxl
- SWIFT_2005_960.mxl
- SWIFT_2005_961.mxl
- SWIFT_2005_962.mxl
- SWIFT_2005_963.mxl
- SWIFT_2005_964.mxl
- SWIFT_2005_965.mxl
- SWIFT_2005_966.mxl
- SWIFT_2005_967.mxl
- SWIFT_2005_970.mxl
- SWIFT_2005_971.mxl
- SWIFT_2005_972.mxl
- SWIFT_2005_973.mxl
- SWIFT_2005_985.mxl
- SWIFT_2005_986.mxl
- SWIFT_2005_990.mxl
- SWIFT_2005_991.mxl
- SWIFT_2005_992.mxl
- SWIFT_2005_995.mxl
- SWIFT_2005_996.mxl
- SWIFT_2005_998.mxl
- SWIFT_2005_999.mxl
- SWIFT_2006_097.mxl
- SWIFT_2006_101.mxl
- SWIFT_2006_102.mxl
- SWIFT_2006_102_STP.mxl
- SWIFT_2006_103.mxl
- SWIFT_2006_103_STP.mxl
- SWIFT_2006_104.mxl
- SWIFT_2006_105.mxl
- SWIFT_2006_106.mxl
- SWIFT_2006_107.mxl
- SWIFT_2006_110.mxl
- SWIFT_2006_111.mxl
- SWIFT_2006_112.mxl
- SWIFT_2006_190.mxl

- ✦ SWIFT_2006_191.mxl
- ✦ SWIFT_2006_192.mxl
- ✦ SWIFT_2006_195.mxl
- ✦ SWIFT_2006_196.mxl
- ✦ SWIFT_2006_198.mxl
- ✦ SWIFT_2006_199.mxl
- ✦ SWIFT_2006_200.mxl
- ✦ SWIFT_2006_201.mxl
- ✦ SWIFT_2006_202.mxl
- ✦ SWIFT_2006_203.mxl
- ✦ SWIFT_2006_204.mxl
- ✦ SWIFT_2006_205.mxl
- ✦ SWIFT_2006_206.mxl
- ✦ SWIFT_2006_207.mxl
- ✦ SWIFT_2006_210.mxl
- ✦ SWIFT_2006_256.mxl
- ✦ SWIFT_2006_290.mxl
- ✦ SWIFT_2006_291.mxl
- ✦ SWIFT_2006_292.mxl
- ✦ SWIFT_2006_295.mxl
- ✦ SWIFT_2006_296.mxl
- ✦ SWIFT_2006_298.mxl
- ✦ SWIFT_2006_299.mxl
- ✦ SWIFT_2006_300.mxl
- ✦ SWIFT_2006_303.mxl
- ✦ SWIFT_2006_304.mxl
- ✦ SWIFT_2006_305.mxl
- ✦ SWIFT_2006_306.mxl
- ✦ SWIFT_2006_307.mxl
- ✦ SWIFT_2006_308.mxl
- ✦ SWIFT_2006_320.mxl
- ✦ SWIFT_2006_321.mxl
- ✦ SWIFT_2006_330.mxl
- ✦ SWIFT_2006_340.mxl
- ✦ SWIFT_2006_341.mxl
- ✦ SWIFT_2006_350.mxl

- ✦ SWIFT_2006_360.mxl
- ✦ SWIFT_2006_361.mxl
- ✦ SWIFT_2006_362.mxl
- ✦ SWIFT_2006_364.mxl
- ✦ SWIFT_2006_365.mxl
- ✦ SWIFT_2006_380.mxl
- ✦ SWIFT_2006_381.mxl
- ✦ SWIFT_2006_390.mxl
- ✦ SWIFT_2006_391.mxl
- ✦ SWIFT_2006_392.mxl
- ✦ SWIFT_2006_395.mxl
- ✦ SWIFT_2006_396.mxl
- ✦ SWIFT_2006_398.mxl
- ✦ SWIFT_2006_399.mxl
- ✦ SWIFT_2006_400.mxl
- ✦ SWIFT_2006_405.mxl
- ✦ SWIFT_2006_410.mxl
- ✦ SWIFT_2006_412.mxl
- ✦ SWIFT_2006_416.mxl
- ✦ SWIFT_2006_420.mxl
- ✦ SWIFT_2006_422.mxl
- ✦ SWIFT_2006_430.mxl
- ✦ SWIFT_2006_450.mxl
- ✦ SWIFT_2006_455.mxl
- ✦ SWIFT_2006_456.mxl
- ✦ SWIFT_2006_490.mxl
- ✦ SWIFT_2006_491.mxl
- ✦ SWIFT_2006_492.mxl
- ✦ SWIFT_2006_495.mxl
- ✦ SWIFT_2006_496.mxl
- ✦ SWIFT_2006_498.mxl
- ✦ SWIFT_2006_499.mxl
- ✦ SWIFT_2006_500.mxl
- ✦ SWIFT_2006_501.mxl
- ✦ SWIFT_2006_502.mxl
- ✦ SWIFT_2006_503.mxl

- SWIFT_2006_504.mxl
- SWIFT_2006_505.mxl
- SWIFT_2006_506.mxl
- SWIFT_2006_507.mxl
- SWIFT_2006_508.mxl
- SWIFT_2006_509.mxl
- SWIFT_2006_510.mxl
- SWIFT_2006_513.mxl
- SWIFT_2006_514.mxl
- SWIFT_2006_515.mxl
- SWIFT_2006_515_SMPG02.mxl
- SWIFT_2006_516.mxl
- SWIFT_2006_517.mxl
- SWIFT_2006_518.mxl
- SWIFT_2006_519.mxl
- SWIFT_2006_524.mxl
- SWIFT_2006_526.mxl
- SWIFT_2006_527.mxl
- SWIFT_2006_528.mxl
- SWIFT_2006_529.mxl
- SWIFT_2006_535.mxl
- SWIFT_2006_536_IOA001.mxl
- SWIFT_2006_536.mxl
- SWIFT_2006_537.mxl
- SWIFT_2006_538.mxl
- SWIFT_2006_540.mxl
- SWIFT_2006_541.mxl
- SWIFT_2006_542.mxl
- SWIFT_2006_543.mxl
- SWIFT_2006_544.mxl
- SWIFT_2006_545.mxl
- SWIFT_2006_546.mxl
- SWIFT_2006_547.mxl
- SWIFT_2006_548.mxl
- SWIFT_2006_549.mxl
- SWIFT_2006_558.mxl

- ◆ SWIFT_2006_559.mxl
- ◆ SWIFT_2006_564.mxl
- ◆ SWIFT_2006_565.mxl
- ◆ SWIFT_2006_566.mxl
- ◆ SWIFT_2006_567.mxl
- ◆ SWIFT_2006_568.mxl
- ◆ SWIFT_2006_569.mxl
- ◆ SWIFT_2006_574_IRSLST.mxl
- ◆ SWIFT_2006_574.mxl
- ◆ SWIFT_2006_574_W8BENO.mxl
- ◆ SWIFT_2006_575.mxl
- ◆ SWIFT_2006_576.mxl
- ◆ SWIFT_2006_577.mxl
- ◆ SWIFT_2006_578.mxl
- ◆ SWIFT_2006_579.mxl
- ◆ SWIFT_2006_581.mxl
- ◆ SWIFT_2006_582.mxl
- ◆ SWIFT_2006_584.mxl
- ◆ SWIFT_2006_586.mxl
- ◆ SWIFT_2006_587.mxl
- ◆ SWIFT_2006_588.mxl
- ◆ SWIFT_2006_589.mxl
- ◆ SWIFT_2006_590.mxl
- ◆ SWIFT_2006_591.mxl
- ◆ SWIFT_2006_592.mxl
- ◆ SWIFT_2006_595.mxl
- ◆ SWIFT_2006_596.mxl
- ◆ SWIFT_2006_598.mxl
- ◆ SWIFT_2006_599.mxl
- ◆ SWIFT_2006_600.mxl
- ◆ SWIFT_2006_601.mxl
- ◆ SWIFT_2006_604.mxl
- ◆ SWIFT_2006_605.mxl
- ◆ SWIFT_2006_606.mxl
- ◆ SWIFT_2006_607.mxl
- ◆ SWIFT_2006_608.mxl

◆ SWIFT_2006_609.mxl
◆ SWIFT_2006_643.mxl
◆ SWIFT_2006_644.mxl
◆ SWIFT_2006_645.mxl
◆ SWIFT_2006_646.mxl
◆ SWIFT_2006_649.mxl
◆ SWIFT_2006_690.mxl
◆ SWIFT_2006_691.mxl
◆ SWIFT_2006_692.mxl
◆ SWIFT_2006_695.mxl
◆ SWIFT_2006_696.mxl
◆ SWIFT_2006_698.mxl
◆ SWIFT_2006_699.mxl
◆ SWIFT_2006_700.mxl
◆ SWIFT_2006_701.mxl
◆ SWIFT_2006_705.mxl
◆ SWIFT_2006_707.mxl
◆ SWIFT_2006_710.mxl
◆ SWIFT_2006_711.mxl
◆ SWIFT_2006_720.mxl
◆ SWIFT_2006_721.mxl
◆ SWIFT_2006_730.mxl
◆ SWIFT_2006_732.mxl
◆ SWIFT_2006_734.mxl
◆ SWIFT_2006_740.mxl
◆ SWIFT_2006_742.mxl
◆ SWIFT_2006_747.mxl
◆ SWIFT_2006_750.mxl
◆ SWIFT_2006_752.mxl
◆ SWIFT_2006_754.mxl
◆ SWIFT_2006_756.mxl
◆ SWIFT_2006_760.mxl
◆ SWIFT_2006_767.mxl
◆ SWIFT_2006_768.mxl
◆ SWIFT_2006_769.mxl
◆ SWIFT_2006_790.mxl

- ✦ SWIFT_2006_791.mxl
- ✦ SWIFT_2006_792.mxl
- ✦ SWIFT_2006_795.mxl
- ✦ SWIFT_2006_796.mxl
- ✦ SWIFT_2006_798.mxl
- ✦ SWIFT_2006_799.mxl
- ✦ SWIFT_2006_800.mxl
- ✦ SWIFT_2006_801.mxl
- ✦ SWIFT_2006_802.mxl
- ✦ SWIFT_2006_810.mxl
- ✦ SWIFT_2006_812.mxl
- ✦ SWIFT_2006_813.mxl
- ✦ SWIFT_2006_820.mxl
- ✦ SWIFT_2006_821.mxl
- ✦ SWIFT_2006_822.mxl
- ✦ SWIFT_2006_823.mxl
- ✦ SWIFT_2006_824.mxl
- ✦ SWIFT_2006_890.mxl
- ✦ SWIFT_2006_891.mxl
- ✦ SWIFT_2006_892.mxl
- ✦ SWIFT_2006_895.mxl
- ✦ SWIFT_2006_896.mxl
- ✦ SWIFT_2006_898.mxl
- ✦ SWIFT_2006_899.mxl
- ✦ SWIFT_2006_900.mxl
- ✦ SWIFT_2006_910.mxl
- ✦ SWIFT_2006_920.mxl
- ✦ SWIFT_2006_935.mxl
- ✦ SWIFT_2006_940.mxl
- ✦ SWIFT_2006_941.mxl
- ✦ SWIFT_2006_942.mxl
- ✦ SWIFT_2006_950.mxl
- ✦ SWIFT_2006_960.mxl
- ✦ SWIFT_2006_961.mxl
- ✦ SWIFT_2006_962.mxl
- ✦ SWIFT_2006_963.mxl

- ✦ SWIFT_2006_964.mxl
- ✦ SWIFT_2006_965.mxl
- ✦ SWIFT_2006_966.mxl
- ✦ SWIFT_2006_967.mxl
- ✦ SWIFT_2006_970.mxl
- ✦ SWIFT_2006_971.mxl
- ✦ SWIFT_2006_972.mxl
- ✦ SWIFT_2006_973.mxl
- ✦ SWIFT_2006_985.mxl
- ✦ SWIFT_2006_986.mxl
- ✦ SWIFT_2006_990.mxl
- ✦ SWIFT_2006_991.mxl
- ✦ SWIFT_2006_992.mxl
- ✦ SWIFT_2006_995.mxl
- ✦ SWIFT_2006_996.mxl
- ✦ SWIFT_2006_998.mxl
- ✦ SWIFT_2006_999.mxl
- ✦ SWIFT_2007_097.mxl
- ✦ SWIFT_2007_101.mxl
- ✦ SWIFT_2007_102.mxl
- ✦ SWIFT_2007_102_STP.mxl
- ✦ SWIFT_2007_103.mxl
- ✦ SWIFT_2007_103_STP.mxl
- ✦ SWIFT_2007_104.mxl
- ✦ SWIFT_2007_105.mxl
- ✦ SWIFT_2007_106.mxl
- ✦ SWIFT_2007_107.mxl
- ✦ SWIFT_2007_110.mxl
- ✦ SWIFT_2007_111.mxl
- ✦ SWIFT_2007_112.mxl
- ✦ SWIFT_2007_190.mxl
- ✦ SWIFT_2007_191.mxl
- ✦ SWIFT_2007_192.mxl
- ✦ SWIFT_2007_195.mxl
- ✦ SWIFT_2007_196.mxl
- ✦ SWIFT_2007_198.mxl

- ✦ SWIFT_2007_199.mxl
- ✦ SWIFT_2007_200.mxl
- ✦ SWIFT_2007_201.mxl
- ✦ SWIFT_2007_202.mxl
- ✦ SWIFT_2007_203.mxl
- ✦ SWIFT_2007_204.mxl
- ✦ SWIFT_2007_205.mxl
- ✦ SWIFT_2007_206.mxl
- ✦ SWIFT_2007_207.mxl
- ✦ SWIFT_2007_210.mxl
- ✦ SWIFT_2007_256.mxl
- ✦ SWIFT_2007_290.mxl
- ✦ SWIFT_2007_291.mxl
- ✦ SWIFT_2007_292.mxl
- ✦ SWIFT_2007_295.mxl
- ✦ SWIFT_2007_296.mxl
- ✦ SWIFT_2007_298.mxl
- ✦ SWIFT_2007_299.mxl
- ✦ SWIFT_2007_300.mxl
- ✦ SWIFT_2007_303.mxl
- ✦ SWIFT_2007_304.mxl
- ✦ SWIFT_2007_305.mxl
- ✦ SWIFT_2007_306.mxl
- ✦ SWIFT_2007_307.mxl
- ✦ SWIFT_2007_308.mxl
- ✦ SWIFT_2007_320.mxl
- ✦ SWIFT_2007_321.mxl
- ✦ SWIFT_2007_330.mxl
- ✦ SWIFT_2007_340.mxl
- ✦ SWIFT_2007_341.mxl
- ✦ SWIFT_2007_350.mxl
- ✦ SWIFT_2007_360.mxl
- ✦ SWIFT_2007_361.mxl
- ✦ SWIFT_2007_362.mxl
- ✦ SWIFT_2007_364.mxl
- ✦ SWIFT_2007_365.mxl

✦ SWIFT_2007_380.mxl
✦ SWIFT_2007_381.mxl
✦ SWIFT_2007_390.mxl
✦ SWIFT_2007_391.mxl
✦ SWIFT_2007_392.mxl
✦ SWIFT_2007_395.mxl
✦ SWIFT_2007_396.mxl
✦ SWIFT_2007_398.mxl
✦ SWIFT_2007_399.mxl
✦ SWIFT_2007_400.mxl
✦ SWIFT_2007_405.mxl
✦ SWIFT_2007_410.mxl
✦ SWIFT_2007_412.mxl
✦ SWIFT_2007_416.mxl
✦ SWIFT_2007_420.mxl
✦ SWIFT_2007_422.mxl
✦ SWIFT_2007_430.mxl
✦ SWIFT_2007_450.mxl
✦ SWIFT_2007_455.mxl
✦ SWIFT_2007_456.mxl
✦ SWIFT_2007_490.mxl
✦ SWIFT_2007_491.mxl
✦ SWIFT_2007_492.mxl
✦ SWIFT_2007_495.mxl
✦ SWIFT_2007_496.mxl
✦ SWIFT_2007_498.mxl
✦ SWIFT_2007_499.mxl
✦ SWIFT_2007_500.mxl
✦ SWIFT_2007_501.mxl
✦ SWIFT_2007_502.mxl
✦ SWIFT_2007_503.mxl
✦ SWIFT_2007_504.mxl
✦ SWIFT_2007_505.mxl
✦ SWIFT_2007_506.mxl
✦ SWIFT_2007_507.mxl
✦ SWIFT_2007_508.mxl

- ◆ SWIFT_2007_509.mxl
- ◆ SWIFT_2007_510.mxl
- ◆ SWIFT_2007_513.mxl
- ◆ SWIFT_2007_514.mxl
- ◆ SWIFT_2007_515.mxl
- ◆ SWIFT_2007_515_SMPG02.mxl
- ◆ SWIFT_2007_516.mxl
- ◆ SWIFT_2007_517.mxl
- ◆ SWIFT_2007_518.mxl
- ◆ SWIFT_2007_519.mxl
- ◆ SWIFT_2007_524.mxl
- ◆ SWIFT_2007_526.mxl
- ◆ SWIFT_2007_527.mxl
- ◆ SWIFT_2007_528.mxl
- ◆ SWIFT_2007_529.mxl
- ◆ SWIFT_2007_530.mxl
- ◆ SWIFT_2007_535.mxl
- ◆ SWIFT_2007_536_IOA001.mxl
- ◆ SWIFT_2007_536.mxl
- ◆ SWIFT_2007_537.mxl
- ◆ SWIFT_2007_538.mxl
- ◆ SWIFT_2007_540.mxl
- ◆ SWIFT_2007_541.mxl
- ◆ SWIFT_2007_542.mxl
- ◆ SWIFT_2007_543.mxl
- ◆ SWIFT_2007_544.mxl
- ◆ SWIFT_2007_545.mxl
- ◆ SWIFT_2007_546.mxl
- ◆ SWIFT_2007_547.mxl
- ◆ SWIFT_2007_548.mxl
- ◆ SWIFT_2007_549.mxl
- ◆ SWIFT_2007_558.mxl
- ◆ SWIFT_2007_559.mxl
- ◆ SWIFT_2007_564.mxl
- ◆ SWIFT_2007_565.mxl
- ◆ SWIFT_2007_566.mxl

- ✦ SWIFT_2007_567.mxl
- ✦ SWIFT_2007_568.mxl
- ✦ SWIFT_2007_569.mxl
- ✦ SWIFT_2007_574_IRSLST.mxl
- ✦ SWIFT_2007_574.mxl
- ✦ SWIFT_2007_574_W8BENO.mxl
- ✦ SWIFT_2007_575.mxl
- ✦ SWIFT_2007_576.mxl
- ✦ SWIFT_2007_577.mxl
- ✦ SWIFT_2007_578.mxl
- ✦ SWIFT_2007_579.mxl
- ✦ SWIFT_2007_581.mxl
- ✦ SWIFT_2007_582.mxl
- ✦ SWIFT_2007_584.mxl
- ✦ SWIFT_2007_586.mxl
- ✦ SWIFT_2007_587.mxl
- ✦ SWIFT_2007_588.mxl
- ✦ SWIFT_2007_589.mxl
- ✦ SWIFT_2007_590.mxl
- ✦ SWIFT_2007_591.mxl
- ✦ SWIFT_2007_592.mxl
- ✦ SWIFT_2007_595.mxl
- ✦ SWIFT_2007_596.mxl
- ✦ SWIFT_2007_598.mxl
- ✦ SWIFT_2007_599.mxl
- ✦ SWIFT_2007_600.mxl
- ✦ SWIFT_2007_601.mxl
- ✦ SWIFT_2007_604.mxl
- ✦ SWIFT_2007_605.mxl
- ✦ SWIFT_2007_606.mxl
- ✦ SWIFT_2007_607.mxl
- ✦ SWIFT_2007_608.mxl
- ✦ SWIFT_2007_609.mxl
- ✦ SWIFT_2007_643.mxl
- ✦ SWIFT_2007_644.mxl
- ✦ SWIFT_2007_645.mxl

- ✦ SWIFT_2007_646.mxl
- ✦ SWIFT_2007_649.mxl
- ✦ SWIFT_2007_690.mxl
- ✦ SWIFT_2007_691.mxl
- ✦ SWIFT_2007_692.mxl
- ✦ SWIFT_2007_695.mxl
- ✦ SWIFT_2007_696.mxl
- ✦ SWIFT_2007_698.mxl
- ✦ SWIFT_2007_699.mxl
- ✦ SWIFT_2007_700.mxl
- ✦ SWIFT_2007_701.mxl
- ✦ SWIFT_2007_705.mxl
- ✦ SWIFT_2007_707.mxl
- ✦ SWIFT_2007_710.mxl
- ✦ SWIFT_2007_711.mxl
- ✦ SWIFT_2007_720.mxl
- ✦ SWIFT_2007_721.mxl
- ✦ SWIFT_2007_730.mxl
- ✦ SWIFT_2007_732.mxl
- ✦ SWIFT_2007_734.mxl
- ✦ SWIFT_2007_740.mxl
- ✦ SWIFT_2007_742.mxl
- ✦ SWIFT_2007_747.mxl
- ✦ SWIFT_2007_750.mxl
- ✦ SWIFT_2007_752.mxl
- ✦ SWIFT_2007_754.mxl
- ✦ SWIFT_2007_756.mxl
- ✦ SWIFT_2007_760.mxl
- ✦ SWIFT_2007_767.mxl
- ✦ SWIFT_2007_768.mxl
- ✦ SWIFT_2007_769.mxl
- ✦ SWIFT_2007_790.mxl
- ✦ SWIFT_2007_791.mxl
- ✦ SWIFT_2007_792.mxl
- ✦ SWIFT_2007_795.mxl
- ✦ SWIFT_2007_796.mxl

- ✦ SWIFT_2007_798.mxl
- ✦ SWIFT_2007_799.mxl
- ✦ SWIFT_2007_800.mxl
- ✦ SWIFT_2007_801.mxl
- ✦ SWIFT_2007_802.mxl
- ✦ SWIFT_2007_810.mxl
- ✦ SWIFT_2007_812.mxl
- ✦ SWIFT_2007_813.mxl
- ✦ SWIFT_2007_820.mxl
- ✦ SWIFT_2007_821.mxl
- ✦ SWIFT_2007_822.mxl
- ✦ SWIFT_2007_823.mxl
- ✦ SWIFT_2007_824.mxl
- ✦ SWIFT_2007_890.mxl
- ✦ SWIFT_2007_891.mxl
- ✦ SWIFT_2007_892.mxl
- ✦ SWIFT_2007_895.mxl
- ✦ SWIFT_2007_896.mxl
- ✦ SWIFT_2007_898.mxl
- ✦ SWIFT_2007_899.mxl
- ✦ SWIFT_2007_900.mxl
- ✦ SWIFT_2007_910.mxl
- ✦ SWIFT_2007_920.mxl
- ✦ SWIFT_2007_935.mxl
- ✦ SWIFT_2007_940.mxl
- ✦ SWIFT_2007_941.mxl
- ✦ SWIFT_2007_942.mxl
- ✦ SWIFT_2007_950.mxl
- ✦ SWIFT_2007_960.mxl
- ✦ SWIFT_2007_961.mxl
- ✦ SWIFT_2007_962.mxl
- ✦ SWIFT_2007_963.mxl
- ✦ SWIFT_2007_964.mxl
- ✦ SWIFT_2007_965.mxl
- ✦ SWIFT_2007_966.mxl
- ✦ SWIFT_2007_967.mxl

◆ SWIFT_2007_970.mxl

◆ SWIFT_2007_971.mxl

◆ SWIFT_2007_972.mxl

◆ SWIFT_2007_973.mxl

◆ SWIFT_2007_985.mxl

◆ SWIFT_2007_986.mxl

◆ SWIFT_2007_990.mxl

◆ SWIFT_2007_991.mxl

◆ SWIFT_2007_992.mxl

◆ SWIFT_2007_995.mxl

◆ SWIFT_2007_996.mxl

◆ SWIFT_2007_998.mxl

◆ SWIFT_2007_999.mxl

MX Maps

◆ swiftMX.1-0.camt.003.001.02.mxl

◆ swiftMX.1-0.camt.004.001.02.mxl

◆ swiftMX.1-0.camt.005.001.02.mxl

◆ swiftMX.1-0.camt.006.001.02.mxl

◆ swiftMX.1-0.tsmt.001.001.02.mxl

◆ swiftMX.1-0.tsmt.002.001.02.mxl

◆ swiftMX.1-0.tsmt.003.001.02.mxl

◆ swiftMX.1-0.tsmt.004.001.01.mxl

◆ swiftMX.1-0.tsmt.005.001.01.mxl

◆ swiftMX.1-0.tsmt.006.001.02.mxl

◆ swiftMX.1-0.tsmt.007.001.01.mxl

◆ swiftMX.1-0.tsmt.008.001.02.mxl

◆ swiftMX.1-0.tsmt.009.001.02.mxl

◆ swiftMX.1-0.tsmt.010.001.02.mxl

◆ swiftMX.1-0.tsmt.011.001.02.mxl

◆ swiftMX.1-0.tsmt.012.001.02.mxl

◆ swiftMX.1-0.tsmt.013.001.02.mxl

◆ swiftMX.1-0.tsmt.014.001.02.mxl

◆ swiftMX.1-0.tsmt.015.001.02.mxl

◆ swiftMX.1-0.tsmt.016.001.02.mxl

◆ swiftMX.1-0.tsmt.017.001.02.mxl

◆ swiftMX.1-0.tsmt.018.001.02.mxl

- ✦ swiftMX.1-0.tsmt.019.001.02.mxl
- ✦ swiftMX.1-0.tsmt.020.001.01.mxl
- ✦ swiftMX.1-0.tsmt.021.001.02.mxl
- ✦ swiftMX.1-0.tsmt.022.001.01.mxl
- ✦ swiftMX.1-0.tsmt.023.001.02.mxl
- ✦ swiftMX.1-0.tsmt.024.001.02.mxl
- ✦ swiftMX.1-0.tsmt.025.001.02.mxl
- ✦ swiftMX.1-0.tsmt.026.001.01.mxl
- ✦ swiftMX.1-0.tsmt.027.001.01.mxl
- ✦ swiftMX.1-0.tsmt.028.001.02.mxl
- ✦ swiftMX.1-0.tsmt.029.001.01.mxl
- ✦ swiftMX.1-0.tsmt.030.001.02.mxl
- ✦ swiftMX.1-0.tsmt.031.001.02.mxl
- ✦ swiftMX.1-0.tsmt.032.001.02.mxl
- ✦ swiftMX.1-0.tsmt.033.001.02.mxl
- ✦ swiftMX.1-0.tsmt.034.001.02.mxl
- ✦ swiftMX.1-0.tsmt.035.001.02.mxl
- ✦ swiftMX.1-0.tsmt.036.001.02.mxl
- ✦ swiftMX.1-0.tsmt.037.001.02.mxl
- ✦ swiftMX.1-0.tsmt.038.001.02.mxl
- ✦ swiftMX.1-0.tsmt.039.001.02.mxl
- ✦ swiftMX.1-0.tsmt.040.001.02.mxl
- ✦ swiftMX.1-0.tsmt.041.001.02.mxl
- ✦ swiftMX.1-0.tsmt.042.001.02.mxl
- ✦ swiftMX.2005.camt.007.002.01.mxl
- ✦ swiftMX.2005.camt.008.002.01.mxl
- ✦ swiftMX.2005.camt.026.001.01.mxl
- ✦ swiftMX.2005.camt.027.001.01.mxl
- ✦ swiftMX.2005.camt.028.001.01.mxl
- ✦ swiftMX.2005.camt.029.001.01.mxl
- ✦ swiftMX.2005.camt.030.001.01.mxl
- ✦ swiftMX.2005.camt.031.001.01.mxl
- ✦ swiftMX.2005.camt.032.001.01.mxl
- ✦ swiftMX.2005.camt.033.001.01.mxl
- ✦ swiftMX.2005.camt.034.001.01.mxl
- ✦ swiftMX.2005.camt.035.001.01.mxl

- ✦ swiftMX.2005.camt.036.001.01.mxl
- ✦ swiftMX.2005.camt.037.001.01.mxl
- ✦ swiftMX.2005.camt.038.001.01.mxl
- ✦ swiftMX.2005.camt.039.001.01.mxl
- ✦ swiftMX.2-0.semt.001.001.01.mxl
- ✦ swiftMX.2-0.semt.002.001.01.mxl
- ✦ swiftMX.2-0.semt.003.001.01.mxl
- ✦ swiftMX.2-0.semt.004.001.01.mxl
- ✦ swiftMX.2-0.semt.005.001.01.mxl
- ✦ swiftMX.2-0.semt.006.001.01.mxl
- ✦ swiftMX.2-0.semt.007.001.01.mxl
- ✦ swiftMX.2-0.sese.001.001.01.mxl
- ✦ swiftMX.2-0.sese.002.001.01.mxl
- ✦ swiftMX.2-0.sese.003.001.01.mxl
- ✦ swiftMX.2-0.sese.004.001.01.mxl
- ✦ swiftMX.2-0.sese.005.001.01.mxl
- ✦ swiftMX.2-0.sese.006.001.01.mxl
- ✦ swiftMX.2-0.sese.007.001.01.mxl
- ✦ swiftMX.2-0.sese.008.001.01.mxl
- ✦ swiftMX.2-0.sese.009.001.01.mxl
- ✦ swiftMX.2-0.sese.010.001.01.mxl
- ✦ swiftMX.2-0.sese.011.001.01.mxl
- ✦ swiftMX.2-1.camt.040.001.02.mxl
- ✦ swiftMX.2-1.camt.041.001.02.mxl
- ✦ swiftMX.2-1.camt.042.001.02.mxl
- ✦ swiftMX.2-1.camt.043.001.02.mxl
- ✦ swiftMX.2-1.camt.044.001.01.mxl
- ✦ swiftMX.2-1.camt.045.001.01.mxl
- ✦ swiftMX.2-1.reda.001.001.02.mxl
- ✦ swiftMX.2-1.reda.002.001.02.mxl
- ✦ swiftMX.2-1.reda.003.001.02.mxl
- ✦ swiftMX.2-1.setr.001.001.02.mxl
- ✦ swiftMX.2-1.setr.002.001.02.mxl
- ✦ swiftMX.2-1.setr.003.001.02.mxl
- ✦ swiftMX.2-1.setr.004.001.02.mxl
- ✦ swiftMX.2-1.setr.005.001.02.mxl

- swiftMX.2-1.setr.006.001.02.mxl
- swiftMX.2-1.setr.007.001.02.mxl
- swiftMX.2-1.setr.008.001.02.mxl
- swiftMX.2-1.setr.009.001.02.mxl
- swiftMX.2-1.setr.010.001.02.mxl
- swiftMX.2-1.setr.011.001.02.mxl
- swiftMX.2-1.setr.012.001.02.mxl
- swiftMX.2-1.setr.013.001.02.mxl
- swiftMX.2-1.setr.014.001.02.mxl
- swiftMX.2-1.setr.015.001.02.mxl
- swiftMX.2-1.setr.016.001.02.mxl
- swiftMX.2-1.setr.017.001.02.mxl
- swiftMX.2-1.setr.018.001.02.mxl
- swiftMX.2-2.sese.012.001.01.mxl
- swiftMX.2-2.sese.013.001.01.mxl
- swiftMX.2-2.sese.014.001.01.mxl
- swiftMX.3-0.acmt.001.001.01.mxl
- swiftMX.3-0.acmt.002.001.01.mxl
- swiftMX.3-0.acmt.003.001.01.mxl
- swiftMX.3-0.acmt.004.001.01.mxl
- swiftMX.3-0.acmt.005.001.01.mxl
- swiftMX.3-0.acmt.006.001.01.mxl
- swiftMX.3-1.sese.015.001.01.mxl
- swiftMX.3-1.sese.016.001.01.mxl
- swiftMX.3-1.sese.017.001.01.mxl
- swiftMX.4-0.acmt.001.001.02.mxl
- swiftMX.4-0.acmt.002.001.02.mxl
- swiftMX.4-0.acmt.003.001.02.mxl
- swiftMX.4-0.acmt.004.001.02.mxl
- swiftMX.4-0.acmt.005.001.02.mxl
- swiftMX.4-0.acmt.006.001.02.mxl
- swiftMX.4-0.camt.040.001.03.mxl
- swiftMX.4-0.camt.041.001.03.mxl
- swiftMX.4-0.camt.042.001.03.mxl
- swiftMX.4-0.camt.043.001.03.mxl
- swiftMX.4-0.camt.044.001.02.mxl

- ✦ swiftMX.4-0.camt.045.001.02.mxl
- ✦ swiftMX.4-0.reda.001.001.03.mxl
- ✦ swiftMX.4-0.reda.002.001.03.mxl
- ✦ swiftMX.4-0.reda.003.001.03.mxl
- ✦ swiftMX.4-0.semt.001.001.02.mxl
- ✦ swiftMX.4-0.semt.002.001.02.mxl
- ✦ swiftMX.4-0.semt.003.001.02.mxl
- ✦ swiftMX.4-0.semt.004.001.02.mxl
- ✦ swiftMX.4-0.semt.005.001.02.mxl
- ✦ swiftMX.4-0.semt.006.001.02.mxl
- ✦ swiftMX.4-0.semt.007.001.02.mxl
- ✦ swiftMX.4-0.sese.001.001.02.mxl
- ✦ swiftMX.4-0.sese.002.001.02.mxl
- ✦ swiftMX.4-0.sese.003.001.02.mxl
- ✦ swiftMX.4-0.sese.004.001.02.mxl
- ✦ swiftMX.4-0.sese.005.001.02.mxl
- ✦ swiftMX.4-0.sese.006.001.02.mxl
- ✦ swiftMX.4-0.sese.007.001.02.mxl
- ✦ swiftMX.4-0.sese.008.001.02.mxl
- ✦ swiftMX.4-0.sese.009.001.02.mxl
- ✦ swiftMX.4-0.sese.010.001.02.mxl
- ✦ swiftMX.4-0.sese.011.001.02.mxl
- ✦ swiftMX.4-0.sese.012.001.02.mxl
- ✦ swiftMX.4-0.sese.013.001.02.mxl
- ✦ swiftMX.4-0.sese.014.001.02.mxl
- ✦ swiftMX.4-0.sese.018.001.01.mxl
- ✦ swiftMX.4-0.sese.019.001.01.mxl
- ✦ swiftMX.4-0.setr.001.001.03.mxl
- ✦ swiftMX.4-0.setr.002.001.03.mxl
- ✦ swiftMX.4-0.setr.003.001.03.mxl
- ✦ swiftMX.4-0.setr.004.001.03.mxl
- ✦ swiftMX.4-0.setr.005.001.03.mxl
- ✦ swiftMX.4-0.setr.006.001.03.mxl
- ✦ swiftMX.4-0.setr.007.001.03.mxl
- ✦ swiftMX.4-0.setr.008.001.03.mxl
- ✦ swiftMX.4-0.setr.009.001.03.mxl

- ✦ swiftMX.4-0.setr.010.001.03.mxl
- ✦ swiftMX.4-0.setr.011.001.03.mxl
- ✦ swiftMX.4-0.setr.012.001.03.mxl
- ✦ swiftMX.4-0.setr.013.001.03.mxl
- ✦ swiftMX.4-0.setr.014.001.03.mxl
- ✦ swiftMX.4-0.setr.015.001.03.mxl
- ✦ swiftMX.4-0.setr.016.001.03.mxl
- ✦ swiftMX.4-0.setr.017.001.03.mxl
- ✦ swiftMX.4-0.setr.018.001.03.mxl
- ✦ swiftMX.4-0.setr.047.001.01.mxl
- ✦ swiftMX.4-0.setr.048.001.01.mxl
- ✦ swiftMX.4-0.setr.049.001.01.mxl
- ✦ swiftMX.4-0.setr.050.001.01.mxl
- ✦ swiftMX.4-0.setr.051.001.01.mxl
- ✦ swiftMX.4-0.setr.052.001.01.mxl
- ✦ swiftMX.4-0.setr.053.001.01.mxl
- ✦ swiftMX.4-0.setr.054.001.01.mxl
- ✦ swiftMX.4-0.setr.055.001.01.mxl
- ✦ swiftMX.4-0.setr.056.001.01.mxl
- ✦ swiftMX.4-0.setr.057.001.01.mxl
- ✦ swiftMX.4-0.setr.058.001.01.mxl

Searching for SWIFTNet Correlations

Correlation searches rely on name-value pairs that define the specific data items you are trying to locate. These defined name-value pairs are tracking points for business processes and documents. The SWIFTNet Correlation search feature offers the following additional benefits:

✦ You can receive the results of search queries more quickly.

✦ You can further refine correlation searches by specifying a start and end date/time range.

✦ You can further refine SWIFT correlation searches by specifying SWIFT-specific criteria.

Searching for SWIFT Messages Using Correlations

To perform an advanced search for documents:

1. From the **Administration** menu, select **Business Process** > **Advanced Search** > **SWIFTNet Correlation**.

2. In the **Search SWIFTNet Correlation** section, specify any combination of the following search criteria, as appropriate:

Field	Description	Action
Requestor DN	Distinguished name of the requestor.	Type the name of the requestor.
		Note: This DN must be registered with the SAG instance using SWIFTNet Alliance Webstation.
Responder DN	Distinguished name of the responder.	**Note:** This DN must be registered with the SAG instance using SWIFTNet Alliance Webstation.
Service Name	Name of the service to which both SWIFT correspondents have subscribed.	Type the service name.
		Note: This must be a SWIFTNet service to which you are subscribed.
Request Type	Request type supported by the message exchange.	Type the request type.
Interface	SWIFTNet message type.	Select either InterAct or FileAct.
Operation	The SWIFTNet operation to send an InterAct or FileAct message.	Select from the following:
		◆ get (FileAct)
		◆ put (FileAct)
		◆ sync (synchronous — Interact)
		◆ async (asynchronous — Interact)
SnF	Indicates if the file transfer is done using the store-and-forward method.	Select either True (use Store-and-Forward) or False (do not use Store-and-Forward).
Direction	Indicates the direction of the messages.	Select either Inbound or Outbound.

Field	Description	Action
Status	The status of the messages.	Select one of the following formats: ◆ Success ◆ Failed ◆ In Process
Activity From	Documents in progress or completed after the specified start date and time.	Using the following formats, type a starting date and time range and select **AM** or **PM**: ◆ Date – MM/DD/YYYY ◆ Time – HR:MN:SC
Activity To	Documents in progress or completed before the specified end date and time.	Using the following formats, type an end date and time range and select **AM** or **PM**.: ◆ Date – MM/DD/YYYY ◆ Time – HR:MN:SC
File Name (FileAct Only)	Search for SWIFTNet records that are associated with a specific file.	Type the name of the file for which you want to search the associated records. **Note:** This search is valid for FileAct messages only.
File Size From (FileAct Only)	Search for SWIFTNet records that are associated to files with a specified size range.	Type the beginning size range of the files.
File Size To (FileAct Only)	Search for SWIFTNet records that are associated to files with a specified size range.	Type the endpoint of the range for the file size.
File Size (FileAct Only)	Measurement of file size	Select from the following: ◆ bytes ◆ KB ◆ MB
Results per page	Number of documents you want to display on the results page.	Select the value to indicate the number of documents to display. Valid values are: ◆ 5 ◆ 15 ◆ 30 ◆ 50

3. Click **Go!** The SWIFTNet Correlation page opens, listing the documents that match your search criteria.

Overview of SWIFTNet Transport

The Application enables you to send messages to SWIFTNet using either the InterAct or FileAct protocol. Application enables you to connect to SWIFTNet through the InterAct protocol for real-time messaging, store-and-forward messaging and real-time query and response. It supports delivery notification, non-repudiation, and message priority.

This diagram illustrates the process flow between Application and the SWIFTNet network through the FileAct and InterAct protocols:
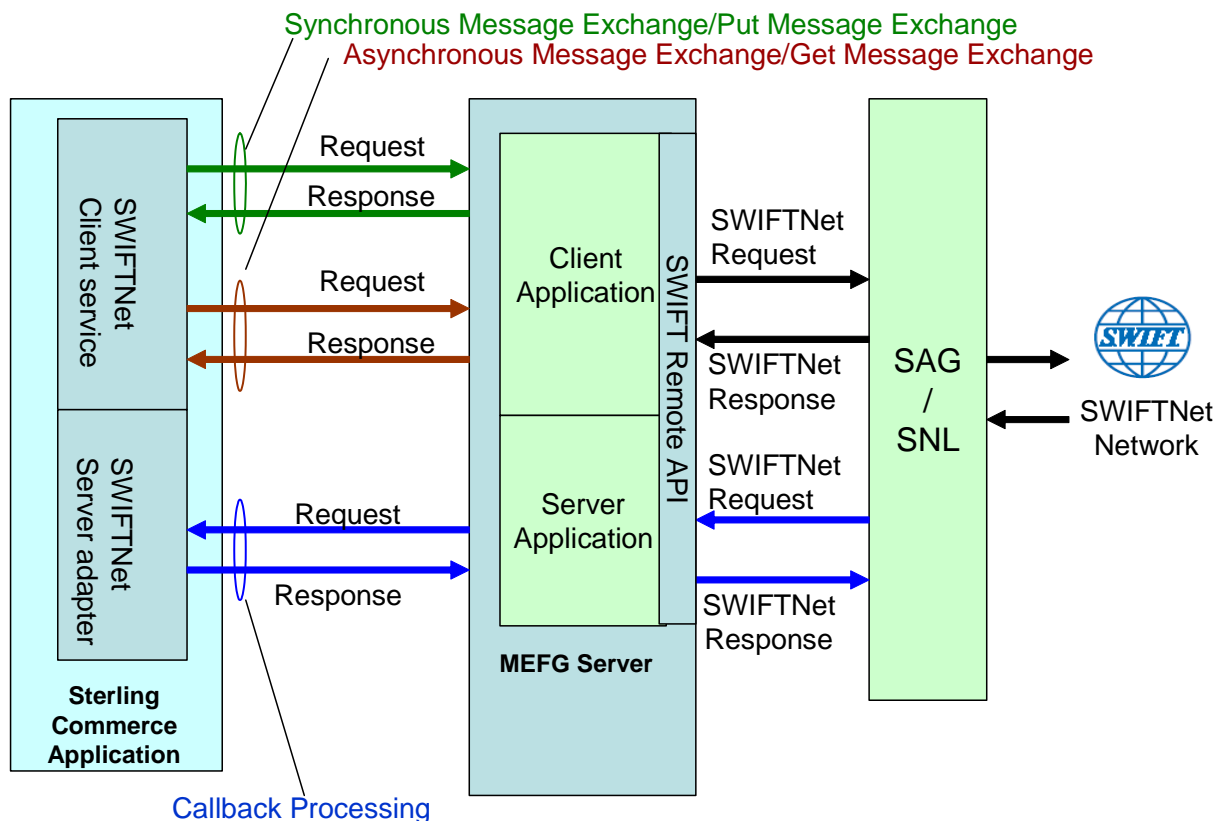
InterAct and FileAct Protocol

When you use the InterAct or FileAct protocol to transport messages to and from SWIFTNet, the SWIFTNet MEFG Server serves requests and receives messages to and from SWIFTNet, through a client application and a server application that communicate with the SWIFTNet network through the InterAct protocol. The SWIFTNet MEFG Server operates independently from Application and includes all the APIs necessary to communicate with the SWIFTNet network.

The SWIFTNet Server adapter is responsible for receiving request messages from SWIFTNet through the SWIFTNet MEFG Server and sending responses back. The SWIFTNet Server adapter is comprised of two parts: the service part and the adapter part. The service part is used in a BPML that does not require

configuration except for enabling it for document tracking. The adapter part is configured through the Admin Console or the GPM, and this adapter is responsible for starting and stopping the SWIFTNet MEFG Server from Application using the Command Line Adapter 2 (CLA2), which is built into the SWIFTNet Server adapter. A Command Line 2 client operates in remote installations (with the SWIFTNet MEFG Server) to enable Application to run a program from a command line in a business process.

The SWIFTNet Client service is responsible for sending SWIFTNet InterAct and FileAct messages (both requests and responses) to SWIFTNet which are initiated by Application. The SWIFTNet Client service enables you to use InterAct and FileAct messaging with a store-and-forward option. The benefits of using store-and-forward include:

✦ The sender and receiver do not need to be online at the same time, as is necessary for real-time messaging.

✦ The sender is notified if a message cannot be delivered (and can optionally be notified when messages are delivered).

The SWIFTNet Client service enables you to use either synchronous or asynchronous messaging using InterAct and Put or Get mode in FileAct.

The SWIFTNet Routing Rule is used by the SWIFTNet Client service to route incoming request to a business process for processing. It uses four parameters:

✦ RequesterDN

✦ ResponderDN

✦ Service name

✦ Request type

These parameters are used to map an incoming request to a business process. The SWIFTNet Routing Rule page enables you to assign any business process to a set of routing parameters. You need to create a SWIFTNet routing rule and associate it with an appropriate business process to process incoming SWIFTNet requests.

Additional FileAct Protocol Options

To send and receive SWIFTNet messages through the FileAct protocol, you will either use the SWIFTNet MEFG Server or use one of the following two methods to connect Application to SWIFT:

✦ **WebSphere MQ Adapter or WebSphere MQ Suite** (see *Configuring the WebSphere MQ Adapter/Suite to Communicate with SWIFT* on page 223)—enables you to configure Application to send and receive InterAct and FileAct files to/from SWIFTNet through the WebSphere MQ Interface for SWIFTAlliance Access (MQSA). Also enables you to send and receive messages to/from SWIFTNet though SWIFTNet Alliance Access (SAA) and the SWIFTNet Remote API Host Adapter (RAHA).

✦ **Connect:Direct for SWIFTNet**—enables you to send and receive FileAct files to/from SWIFTNet through the SWIFTNet Remote API (SWIFTNet RA) and the SWIFTNet Remote API Host Adapter (RAHA).

Application and Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL) is a cryptographic protocol that provides secure communications on the Internet for activities such as web browsing, e-mail, Internet faxing, instant messaging, and other data transfers.

SSL provides endpoint authentication and communications privacy over the Internet using cryptography. When you use SSL, usually only the server is authenticated (the client is not authenticated). The authentication of the server ensures that the end user (which may be a person or an application such as a web browser) knows exactly with whom he or she is communicating. Once the server is authenticated, all the data communicated is encrypted and secured between the client and server only.

Application provides you with the ability to set up SSL in a loopback between the SWIFTNet HTTP Server and the SWIFTNet MEFG Server, so both ends of the communication are secure. You can also initiate or receive an InterAct or FileAct request using SSL (another form of loopback).

CHIPS Message Transport Using SWIFTNet

Application enables you to send CHIPS messages to CHIPS, using either the SWIFTNet network (optionally using IBM Websphere MQ) or The Clearing House Frame Relay Network (a proprietary network that uses IBM Websphere MQ). Acknowledgements are sent to CHIPS using the SWIFTNet Server adapter. Using the SWIFTNet transport available in Application, you can also receive all response messages from CHIPS, including heartbeat messages, and send supervisory STATUS messages to CHIPS to test the connection.

The CHIPS adapter works with the SWIFTNet Server adapter, SWIFTNet HTTP Server adapter, and the SWIFTNet MEFG Server to communicate with CHIPS. When the CHIPS adapter is used with the SWIFTNet network, it receives acknowledgement messages from CHIPS in the SWIFTNet Response within sixty seconds, and any incoming messages (for example, heartbeat message, resolver notification, and so forth) are received by SWIFTNet Server adapter. The return acknowledgement of the incoming messages is performed by the Receive Handler and Acknowledgement Handler within the CHIPS adapter (the business process is bootstrapped using the SWIFTNet Routing Rule).

SWIFTNet MEFG Server

Overview

The Application SWIFTNet MEFG Server serves requests and receives messages to and from SWIFTNet, through a client application and a server application that communicate with the SWIFTNet network through the InterAct or FileAct protocol. The SWIFTNet MEFG Server operates independently from Application and includes all the APIs necessary to communicate with the SWIFTNet network.

Application enables you to use either InterAct or FileAct messaging with a store-and-forward option. The benefits of using store-and-forward include:

✦ The sender and receiver do not need to be online at the same time, as is required for real-time messaging.

✦ The sender is notified in the event delivery fails (and can optionally be notified upon delivery of the message).

Application also has a feature that provides you with failover support from real-time messages to store-and-forward (if there is a failure in real-time messaging, you can configure Application to automatically switch to store-and-forward messaging to increase your messaging success).

Note: This feature requires subscription to both real-time and store-and-forward services.

The SWIFTNet client application sends requests to the SWIFTNet network through the SWIFTNet Alliance Gateway/SWIFTNet Net Link (SAG/SNL) instance. The client application listens for requests from the Application SWIFTNet Client service, and interacts with SWIFTNet to obtain responses.

The SWIFTNet MEFG Server application receives requests from SWIFTNet. The server application listens for requests from SWIFTNet and interacts with Application to obtain responses. A request from the server application to Application calls the SWIFTNet Server adapter to process the request.

The SWIFTNet MEFG Server server application is started by enabling (and stopped by disabling) the Application SWIFTNet Server adapter. The starting and stopping of the server application is handled through the Command Line Adapter 2, which is built into the SWIFTNet Server adapter.

This diagram illustrates the process flow between Application and the SWIFTNet network through the SWIFTNet MEFG Server:
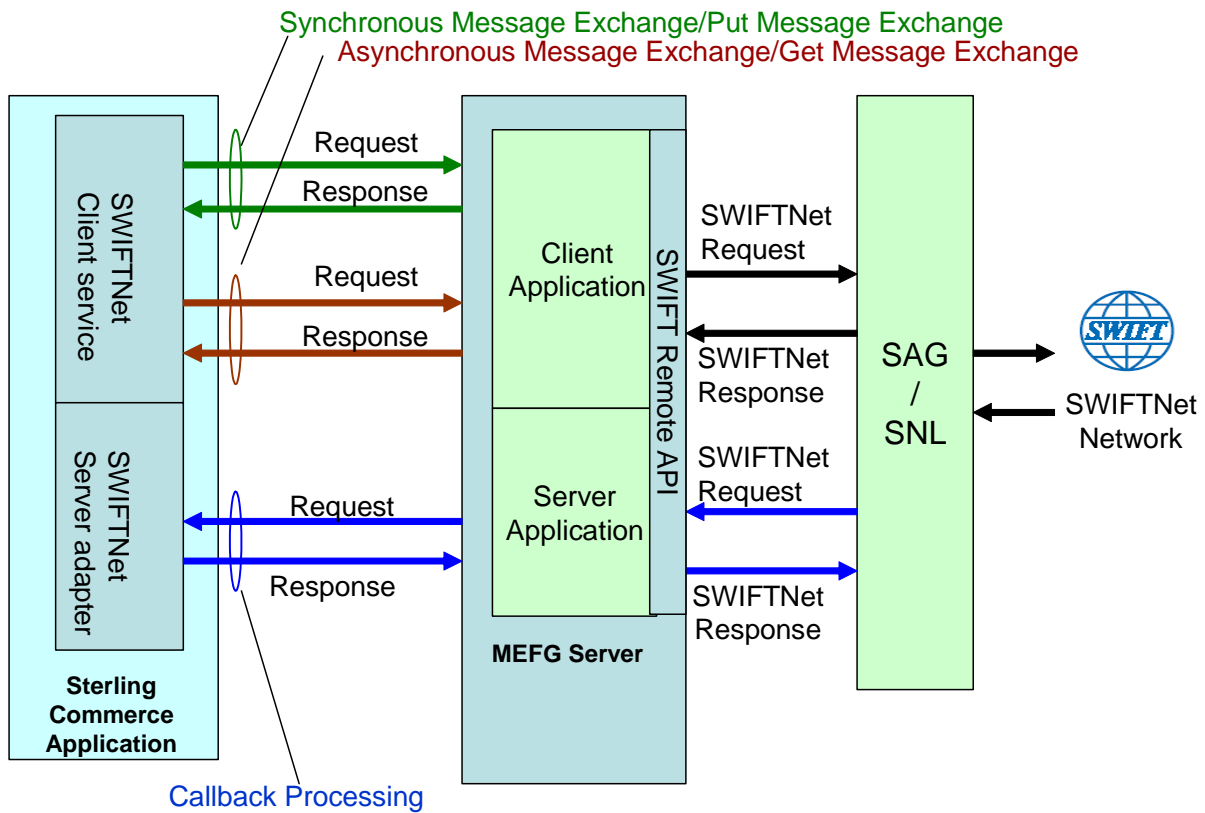
The administration of the SWIFTNet MEFG Server is through SWIFTNet Server adapter, including enabling and disabling the SWIFTNet MEFG Server.

If you use the SWIFTNet HTTP Server adapter in conjunction with the SWIFTNet Server adapter to use Secure Sockets Layer (SSL), the SWIFTNet HTTP Server adapter accepts the forwarded request from the SWIFTNet MEFG Server and provides secure authentication.

This diagram illustrates the process flow between Application and the SWIFTNet network through the SWIFTNet MEFG Server (using the SWIFTNet HTTP Server adapter for SSL):
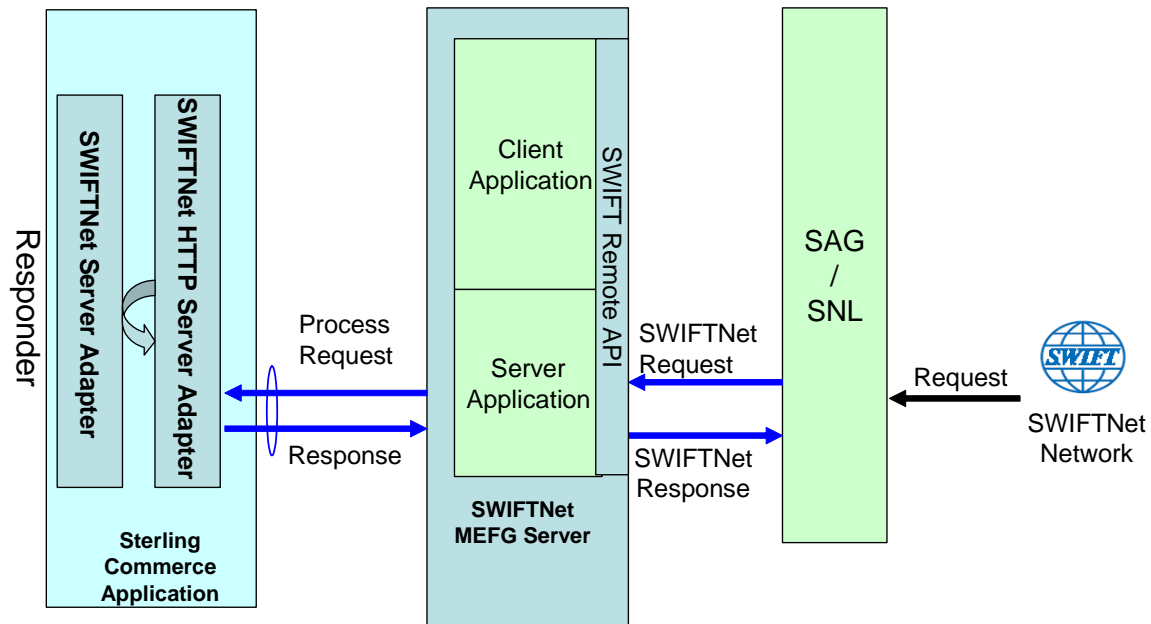
The SWIFTNet Client service (in conjunction with the SWIFTNet HTTP Server adapter) enables you to use Secure Sockets Layer (SSL) to provide secure authentication, using the SWIFTNet HTTP Server adapter to accept the forwarded request from the SWIFTNet MEFG Server. When you use SSL with Application, two channels are secured: an Outbound channel (Application acting as the Requestor) and an Inbound channel (Application acting as the Responder).

This diagram illustrates the process flow between Application and the SWIFTNet network through the SWIFTNet MEFG Server, including the Outbound and Inbound channels (using the SWIFTNet HTTP Server adapter for SSL):
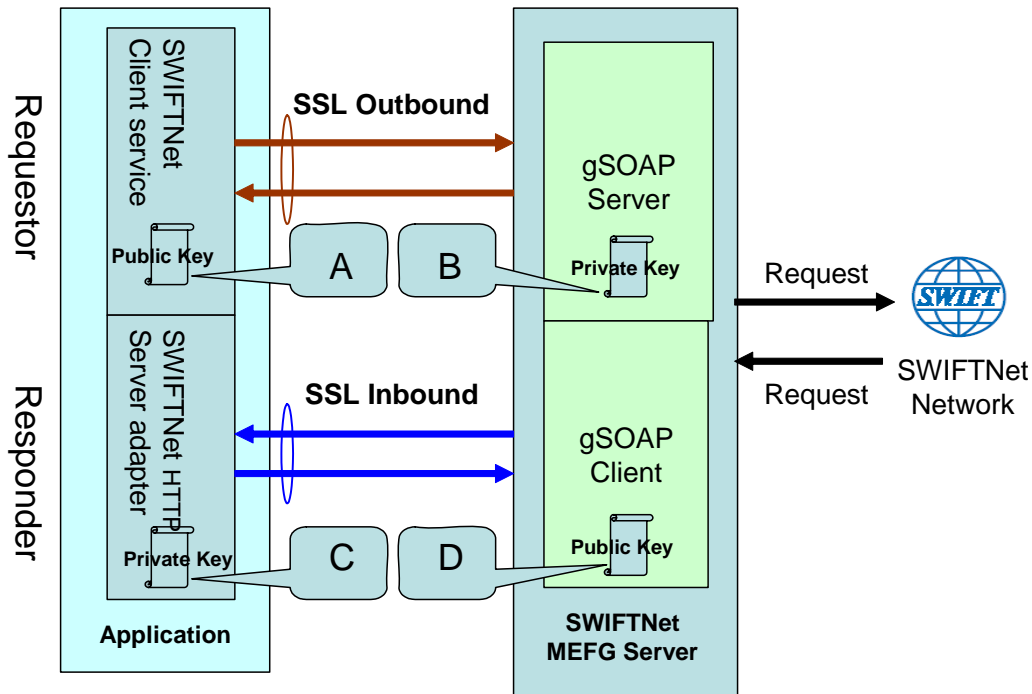
You will need 2 pairs of certificates. The first pair belongs to the SWIFTNet MEFG Server (A and B in the diagram above) and is used to secure the outbound channel. The second pair of certificates belongs to Application (C and D in the diagram above) and is used to secure the inbound channel. In the above diagram, the callouts signify the following:

✦ A — A public key certificate file belongs to the SWIFTNet MEFG Server that is configured on the SWIFTNet Client service (the certificate is specified for the CA Certificate parameter).

✦ B — A private key certificate file that is stored on the SWIFTNet MEFG Server as a key file (which you configure through the SSL Configuration utility named sslUtil.jar in the SWIFTNet MEFG Server installation bin sub-directory).

✦ C — A private key certificate file that is configured on the SWIFTNet HTTP Server adapter (the certificate is specified for the System Cert parameter).

✦ D — A public key file that belongs to Application and is stored for the SWIFTNet MEFG Server as a CA Cert file or trusted list (that you configure through the SSL Configuration utility named sslUtil.jar in the SWIFTNet MEFG Server installation bin sub-directory).

Supported Platforms

The following platforms are supported for the SWIFTNet MEFG Server:

✦ Windows Server 2003 (Standard or Enterprise Edition) with Service Pack 1

✦ AIX 5.3 ML02

Note: SSL is not supported on the AIX 5.2 or 5.3 operating systems for the connection between the SWIFTNet MEFG Server and the following service or adapters: SWIFTNet Client service, SWIFTNet HTTP Server adapter, and SWIFTNet Server adapter. Please note that this does not impact outbound or inbound SSL connectivity between the SWIFTAlliance Gateway (SAG) and SWIFTNet, because secure transmissions to the host are supported.

✦ SunOS 5.10

Client Application

The client application can exchange messages in synchronous or asynchronous mode for InterAct processing or can exchange messages in Put or Get mode for FileAct processing.

Synchronous Message Exchange

When the client application is communicating in synchronous mode messaging, the SWIFTNet Client service prepares the request and sends it to the SWIFTNet MEFG Server. Then, the client application on the SWIFTNet MEFG Server processes the request, performs the necessary communication exchange with the SWIFTNet SAG/SNL instance, and sends the request to the SWIFTNet network.

In synchronous mode, the client application is blocked until a response is received from the responder through the SAG/SNL instance. Once a response is received, it is sent to Application by the client application, and the response is placed in the primary document.

Asynchronous Message Exchange

In asynchronous mode, the SWIFTNet Client service prepares the request and sends it to the SWIFTNet MEFG Server. The client application on the SWIFTNet MEFG Server processes the request, performs the necessary communication exchange with the SWIFTNet SAG/SNL instance, and sends the request to the SWIFTNet network.

In asynchronous mode, the client application receives a response handle from the SAG/SNL instance. Using this response handle, the client application periodically checks with the SWIFTNet network to determine if a response is available . Once a response is received by Application, it is placed in the primary document.

Configuring the Client Application

There is no configuration necessary for the client application (either in synchronous or asynchronous mode or in Put or Get mode), but you must appropriately configure the SWIFTNet Client service to use the client application. See *SWIFTNet Client Service*.

Server Application

When a request from the SWIFTNet network arrives, the SWIFTNet SAG/SNL sends it to the SWIFTNet MEFG Server server application. The server application processes the request and forwards the request to Application. When Application receives the request, it invokes the SWIFTNet Server adapter to process the request. If store-and-forward messaging is in use, the message payload is placed in a mailbox.

Configuring the Server Application

There is no configuration necessary for the server application, but you must appropriately configure the SWIFTNet Server adapter to use the server application. See *SWIFTNet Server Adapter*.

SWIFTNet MEFG Server Installation

The SWIFTNet MEFG Server installation consists of a sequence of related tasks. This table outlines the process flow you must follow to install the SWIFTNet MEFG Server:

Task Number	Description	For more information
1	Configure SAG/SNL.	*Configuring SAG/SNL* on page 153
2	Install and configure the SWIFTNet remote API.	*Installing the SWIFTNet Remote API (RA)* on page 155
3	Install OpenSSL software on the SWIFTNet MEFG Server host. **Note:** Optional—this is only required if you wish to enable SSL support between Application and SWIFTNet MEFG Server.	Obtain the OpenSSL installation software (Win32 OpenSSL v0.9.8e Light version) from: http://www.slproweb.com/products/Win32OpenSSL.html. On Solaris 10 and AIX 5.3 machines, the OpenSSL libraries should be included with your operating system, although they may be optionally installed. Please check the the appropriate documentation for your operating system to ensure the OpenSSL software is installed.
4	Install the SWIFTNet MEFG Server.	*Installing the SWIFTNet MEFG Server* on page 159
5	Configure SSL between Application and the SWIFTNet MEFG Server. **Note:** Optional—this is only required if you wish to enable SSL support between Application and SWIFTNet MEFG Server. **Note:** SSL is not supported on the AIX 5.2 or 5.3 operating systems for the connection between the SWIFTNet MEFG Server and the following service or adapters: SWIFTNet Client service, SWIFTNet HTTP Server adapter, and SWIFTNet Server adapter. Please note that this does not impact outbound or inbound SSL connectivity between the SWIFTAlliance Gateway (SAG) and SWIFTNet, because secure transmissions to the host are supported.	*Configuring SSL Between Application and the SWIFTNet MEFG Server* on page 156
6	Install the Command Line Adapter 2 Client,	*Starting the Command Line Adapter 2 Client* on page 162

Task Number	Description	For more information
7	Configure and enable the SWIFTNet Server Adapter and start the SWIFTNet MEFG Server. **Note:** To monitor the status of the SWIFTNet MEFG Server, you need to select **Show Advanced Status** when you configure the SWIFTNet Server adapter.	*SWIFTNet Server Adapter*
8	Configure and enable the SWIFTNet Client Service.	*SWIFTNet Client Service*
9	Configure the SWIFTNet routing rule.	*SWIFTNet Routing Rule*
10	Configure Mailboxes (only if you are executing store-and-forward).	*Using Mailboxes*
11	Configure the SWIFTNetClient business process.	*SWIFTNetClient Business Process*

Prerequisites

The following prerequisites must be met for the SWIFTNet MEFG Server to operate:

✦ The Command Line Adapter 2 client (CLA2Client) must be running to receive commands from Application to start and stop the SWIFTNet MEFG Server.

✦ The Command Line Adapter 2 client (CLA2Client) must be deployed on the same machine as the SWIFTNet MEFG Server.

✦ SWIFTNet Remote API (RA) must be installed on the same machine as the SWIFTNet MEFG Server.

✦ The SAG/SNL installed and configured with appropriate message partners and endpoints. See *Configuring SAG/SNL* on page 153.

✦ You must have a SWIFTNet Subscription for the InterAct and/or FileAct protocols.

✦ You use the same account to install the SWIFTNet MEFG Server as you used for the SAG/SNL installation.

✦ You have Java JDK 1.5 installed.

✦ You must install the SWIFTNet MEFG Server on either the Sun Solaris 5.10 operating system, Windows Server 2003 (Standard or Enterprise Edition) operating system, or AIX 5.3 operating system.

Configuring SAG/SNL

Complete the following steps to configure SAG/SNL for use with the SWIFTNet MEFG Server:

1. Log in as an administrator to the SWIFTAlliance Workstation.

2. Go to **Gateway Admin - Application Interface** and create the client and server message partners.

   Note: The client (type = Client) and server (type = Server) message partner names must match the names in the SWIFTNet MEFG Server configuration (**<SagMessagePartnerClientName>** and <**SagMessagePartnerServerName>**).

3. In the Application Interface module, for the server message partner, configure the parameters as follows:

Parameter	Configuration
Name	Name from the SWIFTNet MEFG Server configuration (<SagMessagePartnerServerName>).
Type	Server
Status	Enabled
Unit	None
Host Adapter	Remote API Host Adapter
Supported Message Formats	Select **Strict SNL Format**.
Additional Processing	Select **Remote API Host Adapter**.

4. In the Application Interface module, for the client message partner, configure the parameters as follows:

    Note: The Application Interface must be started.

Parameter	Configuration
Name	Name from the SWIFTNet MEFG Server configuration (<SagMessagePartnerClientName>).
Type	Client
Status	Enabled
Unit	None
Default Message Format for Emission (from Message Partner)	Strict SNL Format **Note:** Strict SNL Format is required by the API.
Supported Message Formats	Select **Strict SNL Format**. **Note:** Strict SNL Format is required by the API.
Additional Processing	**Note:** Do not select any additional processing options.

5. In the Endpoints module, for the server message partner, configure the endpoint parameters as follows to define where to route the messages:

Parameter	Configuration
Name	Name from the SWIFTNet MEFG Server configuration.
Destination	Application Interface:<Name from the SWIFTNet MEFG Server configuration>
Status	Enabled

6. In the Endpoints module, for the server message partner, configure the routing detail parameters as follows:

Parameter	Configuration
From	SWIFTNet Interface
Sequence	**Note:** This is the sequence number.
Name	Name from the SWIFTNet MEFG Server configuration (<SNLEndPoint>).
Status	Enabled
SNL Endpoint	None
Service Name	None
Request Type	None
Requestor DN	Relation: **Equals (=)** Parameter: **o=administrator,o=swift**
Responder DN	Relation: **Equals (=)** Parameter: **o=administrator,o=swift**
Traffic Type	None
Delivery Mode	None
Priority	None

7. In the Endpoints module, for the server message partner, configure the destination detail parameters as follows:

Parameter	Configuration
Interface	Application Interface
Application	Name of the SWIFTNet MEFG Server server application (<SagMessagePartnerServerName>).
Mode	Strict

Installing the SWIFTNet Remote API (RA)

You need to install the SWIFTNet Remote API on the machine on which the SWIFTNet MEFG Server will be installed. This is the software distributed by SWIFT, the API that the SWIFTNet MEFG Server uses to connect to the SWIFTNet SAG/SNL instance to link into a SAG.

Complete the following steps to install the RA:

1. Install the remote API on the machine where you will install SWIFTNet MEFG Server.

2. Configure the RA to point to the SAG instance you will be accessing, or if the RA is already configured, verify that it points to the correct SAG instance.

3. If you are installing on the Windows operating system, add **SYSTEM** to the Security settings, allow SYSTEM **Full Control** and select **Allow inheritable permissions from parent to propagate to this object**. See the documentation for the SWIFTNet Remote API for more information.

    a. Right-click and select **Properties** on <SWIFT RA API installdir>.

    b. Select the **Security** tab.

    c. Click **Add** to and select **SYSTEM**.

    d. Allow **Full Control** to **SYSTEM**.

    e. Select the **Allow inheritable permissions from parent to propagate this object** option.

    f. To confirm, navigate to the <Applicationinstalldir\SWIFTAlliance\RA\lib> directory, and right-click and select **Properties**.

    g. Select the **Security** tab.

    h. Verify that SYSTEM has Full Control permissions for the <Applicationinstalldir\SWIFTAlliance\RA\lib> directory.

4. If you are installing on a Windows operating system, following the SWIFT RA installation, you will need to set the PATH system environment variables:

```
PATH : append <Swift RA API installdir>\bin;<Swift RA API installdir>\lib
SWNET_HOME : <Swift RA API installdir>
SWNET_CFG_PATH : <Swift RA API installdir>\Ra1\cfg;
```

Configuring SSL Between Application and the SWIFTNet MEFG Server

Note: Optional—this is only required if you wish to enable SSL support between Application and SWIFTNet MEFG Server.

To configure SSL for the SWIFTNet MEFG Server, you must complete the following tasks:

Task Number	Descripton
1	Prepare the SSL certificates for the SWIFTNet MEFG Server.
2	Prepare the SSL certificates for Application.
3	Configure the SWIFTNet Server adapter.
4	Configure the SWIFTNetClient business process.
5	Configuring the SSL Setup on the SWIFTNet MEFG Server

Preparing the SSL Certificates for the SWIFTNet MEFG Server

To prepare the SSL certificates for use with the SWIFTNet MEFG Server, complete the following:

1. Create the keyfile that contains the private key for MEFGCommServer.

    You can use OpenSSL to generate the keyfile and certificate request file. Then, you can use this certificate request to ask the CA to generate the certificate and sign it for you.

Note: Note: When you use OpenSSL to generate the keyfile, you will be prompted to type in the password to protect the keyfile.

Note: Please take note of this password for later use. (step 8 of *Configuring the SSL Setup on the SWIFTNet MEFG Server*).

2. Import the signed certificates into the Application CA repository.

3. Note the CA Certificate ID and CA Certificate Name because you will need to use them in the SWIFTNetClient business process.

4. Create the CA Certificate that contains the cert (public key) for the Application side of this configuration.

Note: You can only do this after you have completed step 3 of *Preparing the SSL Certificates for Application*.

Preparing the SSL Certificates for Application

To prepare the SSL certificates for use with Application, complete the following:

1. Create a self-signed certificate on Application for the System certificate, including the following:

   ◆ Select **Set Certificate Signing Bit**.

   ◆ The name of this certificate must be the name of your server/domain so that Open SSL can properly validate the certificate.

Note: Alternatively, you can generate a certificate signing request using the Application Certificate Wizard, and ask a CA to sign your certificate. If you choose this option, include the following:

   ◆ Ensure your common name for the certificate matches the correct server name/domain name.

   ◆ Check in the key and the certificate (after the certificate is signed and returned by CA) to Application System Certificate.

2. When you create and check in the certificate, note the certificate name. This name will be used when you configure the SWIFTNet HTTP Server adapter (step 4 below).

3. Export the public key of the certificate you generated above. This public key will be used by the SWIFTNet MEFG Server (that you created in *Preparing the SSL Certificates for the SWIFTNet MEFG Server* on page 156) as CA certificate file (trusted list).

4. Configure the SWIFTNet HTTP Server adapter to use SSL and choose the System certificate you generated in step 1 of this procedure.

Note: Note the port number of the SWIFTNet HTTP Server adapter because this number must match the port you configure for the SWIFTNet Server adapter (below).

Configuring the SWIFTNet Server Adapter for SSL

To configure the SWIFTNet Server adapter for SSL, complete the following:

1. Select **Deployment > Services > Configuration**.

2. Search for SWIFTNet Server adapter or select it from the list and click **Go!**.

3. Click **Edit**.

4. For **GIS HTTP Server Adapter Port**, use the SSL port configured for the SWIFTNet HTTP Server adapter (above).

5. For **GIS Server IP**, type your exact server name/domain name. This must match with the server/domain name and also match the system certificate name you created in *Preparing the SSL Certificates for Application* on page 157.

6. Ensure that **Use SSL** is set to **True**.

Configuring the SWIFTNet HTTP Server Adapter for SSL

To configure the SWIFTNet HTTP Server adapter for SSL, complete the following:

1. Select **Deployment > Services > Configuration**.

2. Search for SWIFTNet HTTP Server adapter or select it from the list and click **Go!**.

3. Click **Edit**.

4. Ensure that **Use SSL** is set to **Must**.

5. For **System Certificate**, select the appropriate system certificate.

Configuring the SWIFTNet Client Service or Business Process for SSL

You must either configure the SWIFT Client service through the Application user interface or through the business process you create for the service.

To configure the SWIFTNet Client service for SSL, complete the following:

1. Select **Deployment > Services > Configuration**.

2. Search for SWIFTNet Client service or select it from the list and click **Go!**.

3. Click **Edit**.

4. Ensure that **Use SSL** is set to **Must**.

5. For **CA Certificate**, select the appropriate CA certificate. This is the certificate you imported in step 2 of *Preparing the SSL Certificates for the SWIFTNet MEFG Server* on page 156.

Alternatively, to configure the SWIFTNetClient business process, add the following to the BPML to ensure the SSL configuration for the SWIFTNet HTTP Client adapter is included:

Note: The **bold** lines indicate information that you need to modify to match your installation.

```
<<operation>
      <participant name="SWIFTNetClientService"/>
      <output message="handleClientRequest">
        <assign to="." from="*"></assign>
        <assign to="interfaceMode">interact</assign>
        <assign to="swiftOp">sync</assign>
        <assign to="requestorDN">o=ptscfrnn,o=swift</assign>
        <assign to="responderDN">o=ptscfrnn,o=swift</assign>
        <assign to="serviceName">swift.generic.ia!x</assign>
        <assign to="SnF">FALSE</assign>
        <assign to="nonRepudiation">FALSE</assign>
        <assign to="possibleDuplicate">FALSE</assign>
        <assign to="deliveryNotification">FALSE</assign>
        <assign to="UseSSL">TRUE</assign>
```

```
        <assign to="CipherStrength">All</assign>
 <assign to="CACertId">000.00.000.00:00000:10f3202f455:4337</assign>
     </output>
     <input message="testing">
       <assign to="." from="*"></assign>
     </input>
   </operation>
```

Note: The **CACertId** must match the CA Cert ID you configured for the the SWIFTNet MEFG Server (step 2 of *Preparing the SSL Certificates for the SWIFTNet MEFG Server* on page 156).

Configuring the SSL Setup on the SWIFTNet MEFG Server

Prior to completing the next steps, ensure that you have completed the tasks in *Configuring SSL Between Application and the SWIFTNet MEFG Server* on page 156.

1. Type the following command to change to the directory where the SWIFTNet MEFG Server SSL Utilityjar is located:

    ```
    cd <SWIFTNet MEFG Server installdir>/bin
    ```

2. Type the following command to invoke the configuration script:

    ```
    dir *.jar
    ```

    You will see the sslUtil.jar file.

3. Type the following command:

    ```
    java -jar sslUtil.jar
    ```

4. Type the full path for the SWIFTNet MEFG Server home directory (the directory in which you just installed the SWIFTNet MEFG Server) and press **Enter**. You are prompted to confirm the directory.

5. Type **Yes** and press **Enter** to confirm. The configuration script verifies the structure of the directory to ensure that it is the valid installation directory for the SWIFTNet MEFG Server. If the directory is not valid (for example, the bin directory is missing from the path), you are prompted to retype the valid directory. Once a valid directory is entered, you proceed with the SSL configuration.

6. Type the full path to the private key location and press **Enter**. This key file belongs to the SWIFTNet MEFG Server and will used during the SSL "handshake."

Note: The key file contains your private key.

7. Type **Yes** and press **Enter** to confirm the path.

8. Type the correct password to access the key file and press **Enter**.

9. Type the password again to confirm it and press **Enter**.

10. Type the full path to the CA Certificate location and press **Enter**. This CA Certificate file contains the trusted certificates that are used during the SSL "handshake."

11. Type **Yes** and press **Enter** to confirm the path. The configuration completes and displays a message that the SSL configuration process has finished.

Installing the SWIFTNet MEFG Server

Complete the following steps to install the SWIFTNet MEFG Server on a Solarix or AIX system:

Note: The installation script and the binary install are located on the Application installation CD.

1. After installing Application, log on to your UNIX system using the same account as the one you used to install and configure SWIFTNet RA.

2. Download the swifnet_[version].jar file from the installation CD to the server where the SWIFTNet MEFG Server will be installed.

3. Type the following command to invoke the installation script:

   ```
   java -jar swiftnet[version].jar
   ```

   Note: Replace [version] in the command above with the value from the list that matched the patch you are installing.

4. Press **Enter**.

5. Type the destination directory where you want to install the SWIFTNet MEFG Server and press **Enter**.

6. When you are prompted for confirmation, type **y** and press **Enter**.

   Note: To change the destination directory, type **n** and press **Enter**, and repeat step 5. If the destination directory does not have enough free disk space, the script suggests you delete enough files to provide the necessary disk space and then exits the installation.

   The installation script copies files from the distribution media to the destination directory and verifies that the correct number of files and blocks are copied.

   You are notified that the installation is complete with the following message: Installation of Application SWIFTNet MEFG Server component is finished. You will need to configure this application in your Application user interface.

7. If you are installing on the Windows operating system, you are required to enter the account user name in the format **DomainName\Username**. If it is a local user, type **.\Administrator**.

8. Type **Yes** to confirm the account user name and press **Enter**.

9. Type the correct account password, confirm the password, and press **Enter**.

   This installs the following service instances (you can verify this by checking **Control Panel > Administrative tools > Services**):

   - MEFGCommServer Service
   - MEFGSwiftnetServer Service Instance 1
   - MEFGSwiftnetServer Service Instance 2
   - MEFGCommSSLServer Service
   - MEFGSwiftnetSSLServer Service Instance 1
   - MEFGSwiftnetSSLServer Service Instance 2

Note: If you have already installed any of these services, you will be notified through an error message containing Error Code 1073 that the service or services are already installed. If this occurs, you can unregister the services as follows:

    a. Go to <MEFG Server directory>\bin.

    b. Type the following and press **Enter** after each line:

        To uninstall the MEFG Comm Server Service, type `MEFGCommServer.exe -u`

        To uninstall the MEFG SWIFTNet Server Service Instance 1, type `MEFGSwiftnetServer.exe -u s1`

        To uninstall the MEFG SWIFTNet Server Service Instance 2, type `MEFGSwiftnetServer.exe -u s2`

        To uninstall the MEFG Comm SSL Server Service, type `MEFGCommSSLServer.exe -u`

        To uninstall the MEFG SWIFTNet SSL Server Service Instance 1, type `MEFGSWIFTNetSSLServer.exe -u s1`

        To uninstall the MEFG SWIFTNet SSL Server Service Instance 2, type `MEFGSWIFTNetSSLServer.exe -u s2`

10. If you are installing on a Windows operating system, you must add the following under the System variables (add the <MEFG installdir>\bin directory to the PATH environment variable just before the <SWIFT RA API installdir>\bin entry):

    ◆ PATH : append <MEFG installdir>\bin;

Caution: Insert this PATH variable *before* the SWIFT RA API installdir entries you added in step 4 of *Installing the SWIFTNet Remote API (RA)* on page 155 (the PATH variable must appear ahead of the library references in the System variable list).

    ◆ Allow the defined user to start MEFGSwiftnetServer and MEFGCommServer through the following steps:

    a. Select **Control Panel > Administrative Tools > Local Security Settings**.

    b. Select **Local Policies > User Rights Assignment**.

    c. Double-click **Log on as a service** and assign the account user (that you entered during the installation process) to this setting.

11. If you are installing on an AIX operating system, you must modify the startup script **MEFGCommServer.sh** in the **<Swiftnet Server install>/bin** directory by passing option **-o 3**.

In the **MEFGCommServer.sh** startup script, locate this line:

```
${SWV2DIR}/bin/${DAEM_NAME} ${PORT} -s 0 -a ${SWV2DIR} 1>/dev/null
2>/dev/null &
```

And change it by adding the following (in boldface type):

```
${SWV2DIR}/bin/${DAEM_NAME} ${PORT} -o 3 -s 0 -a ${SWV2DIR} 1>/dev/null
2>/dev/null &
```

Configuring Fail-over Processing Using the SWIFTNet MEFG Server

To set up the SWIFTNet MEFG Server in a dual-active SAG configuration for fail-over processing, specify the following application interface definitions for the SWIFTNet Server adapter:

✦ active-active Configuration

✦ RA1 definitions for primary SAG (s1)

✦ RA2 definitions for alternate SAG (s2)

Note: Certificates and profiles must be available on the SAG where they are used. For fail-over processing, Puts and Gets try to connect to the first SAG specified for s1 and s2. If the connection fails, the Put and Get try to connect to the next SAG. If this connection also fails, the cycle is repeated if retry has been enabled.

Starting the Command Line Adapter 2 Client

The Command Line Adapter 2 client (CLA2Client) must be installed and run on a remote server. Complete the following steps to start the remote adapter implementation version of the Command Line Adapter 2:

1. Locate the client jar (CLA2Client.jar) in your Application installation that contains the necessary classes.

2. Move the client jar to the machine that will be running the remote Command Line Adapter 2 client.

   Note: This is the machine on which the SWIFTNet MEFG Server is installed.

3. Start the remote adapter implementation using the following command:

   ```
   [path to java bin]/java -jar [path to CLA2 Client jar file]/CLA2Client.jar
   <port> [debug]
   ```

   Note: The port (above) will be used when you configure the SWIFTNet Server adapter.

   This is an example of the command to start the remote adapter implementation:

   ```
   jdk1.5.0_11/bin/java -jar CLA2Client.jar 15699 debug
   ```

   Note: The [debug] option is not required, but you may find it helpful. When you upgrade Application, you will also need to obtain the corresponding new CLA2Client.jar file to avoid receiving a ClassConflict error.

Monitoring the Status of the SWIFTNet MEFG Server

To monitor the status of the SWIFTNet MEFG Server, you need to select **Show Advanced Status** when you configure the SWIFTNet Server adapter:

1. Select **Deployment > Services > Configuration**.

2. Search for SWIFTNet Server adapter or select it from the list and click **Go!**.

   Note: When you select the SWIFTNet Server adapter, make sure you also select the **Show Advanced Status** check box prior to clicking **Go!**. This enables you to view the Advanced Status column on the Services Configuration page to see whether the SWIFTNet MEFG Server is stopped or started.

3. Click **Edit**.

4. Specify field settings in the Admin Console. See *SWIFTNet Server Adapter*.

5. On the Confirm page, verify that the **Enable Service for Business Processes** check box is selected. This enables the adapter instance.

Starting and Stopping the SWIFTNet MEFG Server

To start and stop the SWIFTNet MEFG Server:

1. Select **Deployment > Services > Configuration**.

2. Search for SWIFTNet Server adapter or select it from the list and click **Go!**.

   Note: When you select the SWIFTNet Server adapter, make sure you also select the **Show Advanced Status** check box prior to clicking **Go!**. This enables you to view the Advanced Status column on the Services Configuration page to see whether the SWIFTNet MEFG Server is stopped or started.

3. Once the SWIFTNet Client adapter is configured and saved, click the **Enabled** check box on the Services Configuration page. This starts the SWIFTNet MEFG Server.

   Note: To stop the SWIFTNet MEFG Server, clear the **Enabled** check box on the Services Configuration page

Document Tracking for SWIFTNet Transport

Overview

The document tracking support within the SWIFTNet Client service and SWIFTNet Server adapter provides you with a document-centric view of the whole process of SWIFTNet messaging. This gives you the ability to monitor the workflow not only from business process point of view, but also from the actual document point of view. In a single view, you can see how the document is transformed/translated from one form to another within Application, and how the request and response document correlate with each other.

To enable this feature, all the business processes that are that are related to the SWIFTNet workflow must have the Document Tracking option enabled when you check in or edit the business processes. To do so and enable the framework to track, select the **Document Tracking** check box on the Process Levels page. Set the following options as needed and leave the rest of the business process parameters as the defaults:

✦ On the **Deadline Settings** page, set the deadline and notification options, if necessary.

✦ On the **Life Span** page, set the life span, if necessary.

Monitoring the SWIFTNet Data Flow

The Data Flow Monitoring feature now enables you to view inbound and outbound SWIFTNet data flows. For both inbound and outbound, you can correlate and view the transformation of the selected document, and you can also see the SWIFTNet Message ID that is related to the document.

Note: Message ID is a unique identifier in SWIFTNet that is required to enable the user to discern duplicate documents—you can tell if a document is a duplicate of another document if the Message IDs of both documents are the same. The Message ID is extremely useful in enabling you to reconcile documents.

To perform an advanced search for SWIFTNet business process data flows:

1. From the **Business Process** menu, select **Advanced Search** > **Data Flows**.

2. In the Business Process Monitor Data Flows page, specify any combination of the following search criteria, as appropriate:

Field	Description
Search	
Endpoint	The remote endpoint of the data flows to search for. Host name or IP address. Optional.
Direction	Direction of the data flows to search for. Optional. Valid values are: ◆ Inbound ◆ Outbound
Protocol	Protocol for the data flows to search for. Optional. Valid values are: ◆ AS2 ◆ HTTP ◆ FTP ◆ SFTP ◆ MBI ◆ Connect:Direct ◆ WebDAV ◆ SWIFTNet
Status	Current or final status of a data flow. Optional. Select one of the following options: ◆ Normal ◆ Error
Document Name	For data flows associated with a specific document, enter the document name. Optional.
Data Size	Range of size of the data transferred to search for. From/To in bytes, KB, MB, or GB. Optional.
DateRange	From – The beginning date and time for data flows to search for To – The end date and time for data flows to search for **Note:** Select the calendar icon to the right of the date to access calendar information. Optional.
Save search results values by using tag	Enter a string for use in repeating the search in another session. Required.

Field	Description
Results per page	Select how many results to display per page. Required. Valid values are:
	◆ 10
	◆ 25
	◆ 50
	◆ 100
	◆ 200
	◆ 250
	◆ 400
	◆ 500
	Default is 10.

List Directly

By Data Flow ID	Type the data flow ID for which you want to search.

3. Click **Go!** The Monitor page opens, listing the business process data flows that match your search criteria.

4. Click the Root Document Name corresponding to the data flow you want to view. There are two different types of data flows:

   ◆ If the Root Document Name is **SWIFTNetRequest**, it corresponds to an outbound data flow in which Application is acting as the client making a SWIFTNet request to the SWIFTNet Alliance Gateway (SAG).

   ◆ If the Root Document Name is **PsHttpDocument_node_***, it corresponds to an inbound data flow in which Application is acting as the server processing the request from the client.

Monitoring the SWIFTNet Communication Session Records

Application creates communication session records for any associated authentication, authorization, file transfer, or non-file transfer records, even if a document is not transferred and no data flow record is created. For example, session data can include a user connecting to a mailbox using FTP, receiving messages, and then quitting the FTP session.

To view SWIFTNet communications sessions records:

1. From the **Administration** menu, select **Business Processes > Advanced Search > Communication Sessions**.

   Complete the fields using the following descriptions:

Field	Description
Endpoint	The remote endpoint of the communication sessions to search for. Host name or IP address. Optional.

Field	Description
Protocol	Protocol for the communication sessions to search for. Optional. Valid values are: ◆ AS2 ◆ HTTP ◆ FTP ◆ SFTP ◆ MBI ◆ Connect:Direct ◆ WebDAV ◆ SWIFTNet
Date Range	From - The beginning date and time to search for communication sessions To - The end date and time to search for communication sessions **Note:** Select the calendar icon to the right of the date to access calendar information. Optional.
Principal	Search for communication sessions associated with a Principal participant. Optional.
Secure Mode	Search for communication sessions in a secure mode. Optional. Valid values are: ◆ SSL ◆ CCC
Save search results values by using tag	Enter a string for use in repeating the search in another session. Required.
Results per page	Select how many results to display per page. Required. Valid values are: ◆ 10 ◆ 25 ◆ 50 ◆ 100 ◆ 200 ◆ 250 ◆ 400 ◆ 500 Default is 10.
List Directly	By Communication Session ID

SWIFT Editor

Application provides you with an online editing interface, the SWIFT Editor, which enables you to correct a SWIFTNet message (MT and MX) that was returned due to an error in translation (a validation check failed) or transmission (including a process failure such as a negative acknowledgement (NAK)). Application maintains a link between every SWIFT message that is returned for reprocessing and its historical predecessor, so there is a record of every rejected message all the way back to the initial submission, including references to each person that modified the message and the return code (error status and reason).

The SWIFT Editor contains a sidebar on the left side that provides useful information about the document history and error reports that includes the reasons for an error that occurred.

The SWIFT Editor also enables you to search through the following code lists quickly and easily:

✦ SWIFT Addresses List
✦ SWIFT Base Addresses List
✦ SWIFT Clearing Code List
✦ SWIFT Currency Code List
✦ SWIFT Country Code List

The editing process requires two different roles (performed by two separate people), per the SWIFT guidelines. Your system administrator will add the appropriate permission (listed in the table below) to the Application user accounts for both roles to enable each role to perform properly.

Role	Description	Application Permission Assigned
Editor	This is the person who edits a SWIFT message.	SWIFT Message Edit
Reviewer	This is the person who submits modified messages to be resent.	SWIFT Message Resend

The combination of both roles provides the "four eyes" validation required by SWIFT.

The editing process is as follows:

1. An e-mail alert is sent to the configured address when an outbound translation has errors during FIN enveloping or when a NAK is received.

2. The Editor searches for the returned message through the EDI Correlations search (searching for messages with ReadyForEdit status), and displays the message in the SWIFT Editor.

3. After the Editor repairs the message (or verifies that it does not need repair) and saves it, a separate e-mail alert is sent to the address configured for the Reviewer (the person responsible for auditing the repair and resending the message).

4. The Reviewer receives the e-mail alert and audits the message (looking for messages with ReadyForResend status). The Reviewer accesses the SWIFT Editor in read-only mode. The error report specifies the original errors and the modified and/or added fields as links to their position in the document.

5. After reviewing a repair message, the Reviewer determines how to handle it. The Reviewer can:

   ◆ Reject changes and mark the message for further edit.

   ◆ Abort the repair process entirely.

   ◆ Resend the message using the specified business process.

Editor Tasks

The Editor repairs and saves SWIFT messages that are returned or rejected because of translation or transmission errors.

The following caveats apply:

✦ The error report displayed in the SWIFT Editor when a Reviewer accesses a repaired document shows all errors and links them to their position in the SWIFT Editor.

✦ All groups and records can be collapsed and expanded as needed to focus viewing.

✦ Highlighting an error icon in the SWIFT Editor displays the error message for reference.

✦ The SWIFT Editor allows users to modify field values and add or delete instances of repeating groups and records.

Editing SWIFT Messages

To find messages in Ready to Edit status (Editor role task) so you can edit them:

1. From the **Administration** menu, select **Business Process** > **Monitor > Advanced Search** > **EDI Correlation**.

2. In the Search Options area, specify the following search criteria, as appropriate:

Field	Description	Action
All Level Options		
Location	EDI correlations maintained in a specific location.	Select **Live Tables** – Display live (active) EDI correlations.
Search Level Type	EDI processing level.	Select **Transaction** – For the search query, display results from the transaction level.

Field	Description	Action
Test Mode	Mode of the Application system where documents that contain the EDI correlations were created.	If desired, select one of the following options: ◆ Any (default) ◆ Test ◆ Production ◆ Information ◆ Interchange is a test ◆ Syntax only test ◆ Echo request ◆ Echo response Optional.
Direction	Flow of the documents that contain the EDI correlations.	If desired, select one of the following options: ◆ Any (default) ◆ Inbound ◆ Outbound Optional.
Sender ID	ID for the organization that is sending documents.	Type the identifier of the sender. Optional.
Receiver ID	ID for the receiving organization.	Type the identifier of the receiver. Optional.
Sender ID Qualifier	Qualifier used with the Sender ID to define the organization that is sending documents.	Type the qualifier of the sender. Optional.
Receiver ID Qualifier	Qualifier used with the Receiver ID for the receiving organization.	Type the qualifier of the receiver. Optional.
Start Date	Documents in progress or completed after the specified start date and time.	Using the following formats, type a starting date and time range and select A.M. or P.M.: ◆ Date – MM/DD/YYYY ◆ Time – HR:MN:SC **Note:** Defaults to a range of the last 24 hours. Optional.
End Date	Documents in progress or completed before the specified end date and time.	Using the following formats, type an end date and time range and select A.M. or P.M.: ◆ Date – MM/DD/YYYY ◆ Time – HR:MN:SC **Note:** Defaults to a range of the last 24 hours. Optional.
Transaction Level Options		

Field	Description	Action
Transaction Set ID	ID of the transaction set indicated in the document.	If desired, type the ID of the transaction set. Optional.
Compliance Status	Status of compliance checking at the transaction set level.	If desired, select one of the following options: ◆ Any (default) ◆ OK ◆ NOT OK Optional.
Message Repair Status	Status of message repair (for SWIFT documents only).	Select **ReadyForEdit** status.

3. Click **Go!** to display the EDI correlation records that match your search criteria.

4. For SWIFT documents, on the EDI Correlation Transaction Results page, click **info** in the Detail column for the document you want to edit.

5. Next to Document Repair Status, select the **ReadyForEdit** link to access that message in the SWIFT Editor.

Note: The Document Repair Status is ReadyForEdit and is a link only if you have the necessary permission to access and edit the document in the SWIFT Editor. If you correct a failed document and save it, the status is changed to ReadyforResend and an e-mail is sent to the address specified in the enveloping.properties property file.

6. Review the errors in the Error Report (left side of the window), select each error link, and repair the error as necessary. This may include changing the content of the field, or adding or deleting fields.

Note: To add an occurrence of a repeating field or group, select **Add** at the appropriate point in the message structure. To delete an occurrence of a field or group, select **Delete** where appropriate.

7. When you are finished editing the message, click **Validate** to validate the message for SWIFT compliance. If the message contains validation errors, the SWIFT editor retains the message so you can correct the errors and perform further edits.

8. Once validation is successful, click **Save**.

9. Review the changes on the Confirm page, and click **Finish**. This sends an e-mail to the Reviewer, notifying that the message has been repaired.

10. In the EDI Correlation Transaction Detail Results page, you can view data details such as message format (MX or MT). If desired, in the EDI Correlation Transaction Detail Results page, click **info** to the right of Document Correlations to get more details about the message.

11. In the Document Correlation Details page, view details about the message you selected, and to see the correlation between the message and corresponding EDI document or data. The details available include:

◆ time stamp

◆ scope

◆ process ID

◆ document name

◆ data value

Note: When you access the returned SWIFT message through the Document Correlation Details page, a tree-view is displayed on the left to allow you to link directly to the previous version(s) of the message. The right pane of the tree view displays the correlation details of the SWIFT message.

12. If you want to view the SWIFT message as text, select the document link at the top right of the page.

Reviewer Tasks

The Reviewer repairs and saves SWIFT messages that are returned or rejected because of translation or transmission errors.

The following caveats apply:

✦ The error report displayed in the SWIFT Editor when a Reviewer accesses a repaired document shows all errors and links to the position of the error in the SWIFT Editor.

✦ All groups and records in the message can be collapsed and expanded (as needed) to focus your viewing.

✦ Highlighting an error icon in the SWIFT Editor Change Report (displayed on the left side of the window) displays the error message for reference.

Searching for SWIFT Messages

To search for messages in Ready to Resend status (Reviewer role task):

1. From the **Administration** menu, select **Business Process** > **Advanced Search** > **EDI Correlation**.

2. In the Search Options area, specify the following search criteria, as appropriate:

Field	Description	Action
All Level Options		
Location	EDI correlations maintained in a specific location.	Select **Live Tables** – Display live (active) EDI correlations.
Search Level Type	EDI processing level.	Select **Transaction** – For the search query, display results from the transaction level.

Field	Description	Action
Test Mode	Mode of the Application system where documents that contain the EDI correlations were created.	If desired, select one of the following options: ◆ Any (default) ◆ Test ◆ Production ◆ Information ◆ Interchange is a test ◆ Syntax only test ◆ Echo request ◆ Echo response Optional.
Direction	Flow of the documents that contain the EDI correlations.	If desired, select one of the following options: ◆ Any (default) ◆ Inbound ◆ Outbound Optional.
Sender ID	ID for the organization that is sending documents.	Type the identifier of the sender. Optional.
Receiver ID	ID for the receiving organization.	Type the identifier of the receiver. Optional.
Sender ID Qualifier	Qualifier used with the Sender ID to define the organization that is sending documents.	Type the qualifier of the sender. Optional.
Receiver ID Qualifier	Qualifier used with the Receiver ID for the receiving organization.	Type the qualifier of the receiver. Optional.
Start Date	Documents in progress or completed after the specified start date and time.	Using the following formats, type a starting date and time range and select A.M. or P.M.: ◆ Date – MM/DD/YYYY ◆ Time – HR:MN:SC **Note:** Defaults to a range of the last 24 hours. Optional.
End Date	Documents in progress or completed before the specified end date and time.	Using the following formats, type an end date and time range and select A.M. or P.M.: ◆ Date – MM/DD/YYYY ◆ Time – HR:MN:SC **Note:** Defaults to a range of the last 24 hours. Optional.
Transaction Level Options		

Field	Description	Action
Transaction Set ID	ID of the transaction set indicated in the document.	If desired, type the ID of the transaction set. Optional.
Compliance Status	Status of compliance checking at the transaction set level.	If desired, select one of the following options: ◆ Any (default) ◆ OK ◆ NOT OK Optional.
Message Repair Status	Status of message repair (for SWIFT documents only).	Select **ReadyForResend** status.

3. Click **Go!** to display the EDI correlation records that match your search criteria.

4. For SWIFT documents, on the EDI Correlation Transaction Results page, click **info** in the Detail column for the document you want to audit.

5. Next to Document Repair Status, select the **ReadyForResend** link to display the SWIFT Editor.

Note: The Document Repair Status is ReadyForResend and is a link only if you have the necessary permission to audit the document in the SWIFT Message Editor.

6. Review the changes made in the Change Report (left side of the window), select each link, and review the change as necessary.

7. When you are finished auditing the entire message, click **Next**.

8. Specify how you want to handle this message by performing one of the following:

    ◆ Mark this document for further edit (sends the message back to the Editor for further editing).

    ◆ Resend this document (sends the document using the business process you select).

    ◆ Abort (terminates the audit process).

9. If you are resending the message, select the appropriate business process from the **Execute Business Process** list. By default it is the SWIFTEnvelope business process.

10. If you want to disable validation prior to resending the message, select **Turn Off Validation**.

Note: Select this option if you want to send a message that does not conform to SWIFT validation rules.

11. Review the Confirm page, and click **Finish**.

SWIFT Message Entry Workstation

Application provides you with the SWIFT Message Entry Workstation functionality that enables you to create, edit, review, copy, delete, and track a SWIFT message.

The SWIFT Message Entry Workstation enables you to create a new SWIFTNet message (MT or MX), and to enter the message data based on the message structure. Additionally, the SWIFT Message Entry Workstation provides validation functionality that flags syntactical errors that may need to be fixed before sending the message. The SWIFT Message Entry Workstation also allows you to repair errors in the message.

The extent of functionality that is available to each user of the SWIFT Message Entry Workstation is determined by the security permissions that your system administrator assigns to your user account, based on your role. In Application, each user account has permissions assigned to it, so that the user can perform necessary functions (for SWIFT Message Entry Workstation, this means that you have the required permissions to perform necessary actions for SWIFT messages, depending on your role). The message creation process requires three different roles (performed by three separate people as described in the table below):

Role	Description	Permissions that Must be Assigned to the User Account
Creator	This is the person who creates a message. The Creator can also validate a message.	◆ Message Entry Create ◆ Message Entry Edit ◆ Message Entry Delete ◆ Message Entry Copy
Verifier	This is the person who can submit the created or edited messages to be sent. The Verifier can also validate the messages.	Message Entry Review
Tracker	This is the person who tracks the history of the messages.	Message Entry Track

Each SWIFTNet message created through the SWIFT Message Entry Workstation evolves in stages. The possible statuses of a SWIFT message during this evolution are as follows:

✦ Draft (this is the initial stage and subsequent edits)

✦ Ready to send

✦ Sent

✦ Rejected

✦ Deleted

SWIFT Message Creation Process

The message creation process is as follows:

Task	Process
1	The Creator logs in to Application and the user permissions assigned to the Creator are loaded.
2	Using the SWIFT Message Entry Workstation, the Creator creates a new SWIFT message or searches for a SWIFT message which is in Draft state for further editing.
3	After the Creator creates the message, he or she can validate the message to check for SWIFT compliance and mark the message as **Ready to Send**.
4	The Verifier reviews the message and can also perform the following tasks: ◆ Reject the message and mark the message for further edit, including the reasons for rejection (if desired). ◆ Send the message, which invokes the business process (SWIFTMessageEntryEnvelope is the default). Once the business process is invoked, the scope of Message Entry Workstation ends. **Note:** If you want to disable validation prior to sending the message, select **Turn Off Validation**.
5	If the message is rejected by the Verifier, the Creator can edit the changes using the SWIFT Editor and then mark the message as **Ready for Resend**, and the Verifier can again mark it as **Reject** or **Send**.
6	The Tracker can track the history of the message and review the modifications that were made to the message, including the user who made each change (along with any remarks noted about why a change was made). Also the tracker can view the raw data for the message, regardless of the message status.

Creating a SWIFT Message

To create new messages in the SWIFT Message Entry Workstation (Creator Role Task):

1. From the **Administration** menu, select **Business Process** > **Message Entry Workstation**.

2. Under Create, next to New Message, click **Go!**.

Note: The New Message option is displayed only if you have the Message Entry Create permission added to your Application user account.

3. In the **Select New Message** page, enter the following details, as required, and click **Next**:

Field	Description
Message name	The name of the SWIFT message. Required.
Sender ID	Identifier for the organization that is sending this SWIFT message. Optional. **Note:** The Sender ID value is required for some SWIFT messages since this value is used to perform validation against your data.

Field	Description
Receiver ID	Identifier for the organization receiving this SWIFT message. Optional. **Note:** The Receiver ID value is required for some SWIFT messages since this value is used to perform validation against your data.

4. In the **Select Message Information** page, enter the following details (as required), and click **Next**.

Field	Description
Standard	The standard required to create the SWIFT message.
Standard Version	The version of the standard required to create the message.
Message Type	The type of message being created.

5. In the Confirm page, click **Finish**. This launches the **SWIFT Editor** to enable you to type the values in the required parameters to create the new SWIFT message.

Note: Any data typed for the **Optional Repetitive Group** is displayed under the **Added Groups** section in the Confirmation page. To add an occurrence of a repeating field or group, click **Add** at the appropriate point in the message structure. To delete an occurrence of a field or group, click **Delete** where appropriate.

6. In the SWIFT Editor page, the **Validate** function enables you to validate the message for SWIFT compliance so that you can avoid translation errors during enveloping.

7. If any errors occur during validation, review the errors in the Validation Report (left side of the window), select each error link, and repair the error as necessary (repairing may include changing the content of the field, or adding or deleting fields).

8. You can either save the message for further edits by selecting the SAVE option or submit the message for review by selecting the READY TO SEND option. When the action is completed successfully, you are prompted with the following message: "The system update completed successfully." Once a message status is Ready to Send, the message can be reviewed by the Verifier.

9. An e-mail alert is sent to the Verifier when the status of the message is changed from DRAFT to READY TO SEND. The e-mail address of the Verifier is specified in the **messageentryworkstation.properties.in** property file. Any changes to this e-mail address can be done by editing the customer_overrides.properties file in the following format: **messageentry.SWIFT.READY_TO_SEND.<username>** = **e-mail** address of the verifier, where **username** is the user who marks the messages as READY TO SEND.

Note: If there is no e-mail address specified in the entry **messageentry.SWIFT.READY_TO_SEND.<username>** the system searches for the e-mail address specified in the entry **messageentry.SWIFT.READY_TO_SEND.default**. All ReadyToSend messages will be directed to this e-mail address. By default this is the e-mail address of the system administrator.

Searching for a SWIFT Message

SWIFT Message Entry Workstation enables you to easily search for messages as follow:

1. From the **Administration** menu, select **Business Processes > Message Entry Workstation**.

The parameters described in the Search Parameters table are displayed.

Search Parameters

The following table describes the search parameters displayed on the Message Entry Workstation page.

Field	Description	Action
Message Name	The name used for the SWIFT message.	Type the name of the SWIFT message for which you are searching.
Standard	The standard used for the SWIFT message.	Select the SWIFT standard used for creating the message.
Standard Version	The version used for creating the SWIFT message.	Select the standard version used for the SWIFT message.
Message Type	The type of message being created.	Select the type of message that was used to create the message.
Sender ID	Identifier for the organization that is sending the SWIFT message.	Type the identifier of the sender.
Receiver ID	Identifier for organization receiving the SWIFT message.	Type the identifier of the receiver.
Author	The user who created the message.	Type the name of the person who created the message.
Message Status	Status of the message.	Select the status of the message from the following options: ◆ DRAFT ◆ READY TO SEND ◆ SENT ◆ REJECTED
Start Date	Documents in progress or completed after the specified start date and time.	Using the following formats, type a starting date and time range and select A.M. or P.M.: ◆ Date – MM/DD/YYYY ◆ Time – HR:MN:SC **Note:** Defaults to a range of the last 24 hours.

Field	Description	Action
End Date	Documents in progress or completed before the specified end date and time.	Using the following formats, type an end date and time range and select A.M. or P.M.: ◆ Date – MM/DD/YYYY ◆ Time – HR:MN:SC **Note:** Defaults to a range of the last 24 hours.

2. Depending on the search criteria, the Search Result page displays various parameters as follows:

Search Results Parameters

Field	Description
Select	Enables the Creator to copy, edit, delete, or get information of a message.
Name	Displays the name of the message.
Standard	Displays the SWIFT message standard.
Version	Displays the version of the SWIFT standard.
Type	Displays the type of SWIFT message.
Sender ID	Identifier for the organization that is sending the SWIFT message.
Receiver ID	Identifier for the organization receiving the SWIFT message.
Created On	Displays the date and time when the message was created.
Status	Displays the status of the message.

Note: If there are no messages found using the search criteria you specify, the message "No Messages match your search criteria" is displayed.

3. Depending on the results displayed for the message for which you searched, you can click **copy**, **edit**, **delete**, or **info** from the Select Field for that SWIFT message.

4. Click **Return** to return to the main page of the SWIFT Message Entry Workstation.

Verifying SWIFT Messages

The Verifier may not create messages using the SWIFT Message Entry Workstation, but may only verify and validate existing messages.

The following caveats apply to the Verifier role:

✦ The Verifier uses the SWIFT Editor in read-only mode.

✦ All groups and records in the message can be collapsed and expanded (as needed) so you can focus your viewing of specific parts of the message.

✦ Clicking **info** in the Change Report (displayed on the left side of the SWIFT Editor page) displays the details of the changes made for reference.

To search a message, the verifier performs the following actions:

1. From the **Administration** menu, select **Business Process** > **Message Entry Workstation**.

2. The Verifier searches for a SWIFT message based on the search criteria described in *Search Parameters* on page 178.

3. After searching for a SWIFT message (using the procedure for searching a message as described in the section *Searching for a SWIFT Message* on page 178), select it and click **Send**. The SWIFT Editor is displayed in read-only mode.

4. When you are finished auditing the entire message, click **Next**.

5. In the Confirm page, specify how you want to handle this message by performing one of the following options:

   ◆ Select the action **Send** to invoke a business process. The default business process is SWIFTMessageEntryOutbound). This business process looks up the envelope that is used to send the message.

Note: If during the message creation process, the Sender ID and Receiver ID were specified for the message, the values for those parameters are automatically populated to these fields and you are unable to modify the values.

   ◆ Select the action **Reject** (optionally, you can provide reason for rejection). The **Reject** action updates the message to Rejected status. For each Rejected message, the Creator must make the necessary edits prior to marking the document **Ready for Resend**.

Note: By default, the business process used for Execute Business Process is **SWIFTMessageEntryOutbound,** which is specified in the **messageentryworkstation.properties.in** property file. If you want to specify a different business process, you must add the name of that business process to the customer_overrides.properties file in the following format: envelope.messageentry=NameOfBusinessProcess (where NameOfBusinessProcess is the name of the business process).

6. Review the Confirm page and click **Finish**.

Note: The Verifier can only access documents with Ready to Send status. The Verifier is not able to take any action on a message with any other status.

7. An e-mail alert is sent when the status of the message is changed from READY TO SEND to REJECTED. The e-mail alert is sent to the user who last modified the message (not necessarily the user who created it). An e-mail address is specified for each user when the system administrator creates the user accounts.

Note: If no e-mail address is specified for the user who last modified the message, the system checks for the e-mail address specified in the **messageentryworkstation.properties.in** property file. If neither e-mail address is specified, the system searches for the e-mail address specified in the entry **messageentry.SWIFT.REJECT.default** which is available in the **messageentryworkstation.properties.in** property file. By default, this is the e-mail address of the system administrator. You can change this e-mail address by editing the customer_overrides.properties file in the following format:

messageentry.SWIFT.<username>.mailid = **e-mail address** of the Creator or the person who last edited the message, where **username** is the user who marks the message as REJECTED.

Tracking SWIFT Messages

A Tracker can track the history of a message and view the changes made by each user. The Tracker can also view the raw data for the different message statuses.

In order to track a message, the Tracker must perform the following actions:

1. From the **Administration** menu, select **Business Process** > **Message Entry Workstation**.
2. The Tracker searches for a message based on the search criteria as described in the *Search Parameters* on page 178.

   Depending on the search criteria you specify, the Search Result page displays various parameters as described in *Search Results Parameters* on page 179.
3. In the Message page, click **info** next to the appropriate message to display the following details:

Field	Description
Document	Click **info** to view the raw data for the message.
Modified on	The date the message was modified.
Modified by	The user who modified the message.
Status	The status of the message after modification.
Remarks	Remarks made by a user or by the system, if any.

Deleting a Message

The SWIFT Message Entry Workstation allows you to delete a message if you have the appropriate permission assigned to your user account. Both the Creator and the Verifier are able to delete messages.

To delete a message, you must perform the following actions:

1. From the **Administration** menu, select **Business Process** > **Message Entry Workstation**.
2. Search for a message based on the search criteria described in *Search Parameters* on page 178.

   Depending on the search criteria you specify, the Search Result displays various parameters as described in *Search Results Parameters* on page 179.

Note: You must select the **Delete** function from the Select menu in the same row as the message that you want to delete. You are prompted with a delete confirmation window.

3. Click **OK** to delete the message.

4. In the Delete Resources page, review the following message details to ensure you are deleting the right message and click **Next**:

Field	Description
Message Name	Displays the name of the SWIFT message that you want to delete.
Standard	Displays the SWIFT message standard.
Standard Version	Displays the version of the SWIFT standard.
Message Type	Displays the type of SWIFT message.
Created On	Displays the date and time when the message was created.
Created By	Displays the name of the user who created the message.
Current Status	Displays the current status of the SWIFT message.

5. In the Confirm page, click **Delete**. When the deletion is complete, the message "the system update completed successfully" is displayed.

6. Click **Return** to return to the main page of the SWIFT Message Entry Workstation.

Copying a Message

The SWIFT Message Entry Workstation allows you to copy a message if you have the Message Entry Copy permission assigned to your user account.

To copy a message, you must perform the following actions:

1. From the **Administration** menu, select **Business Process** > **Message Entry Workstation**.

2. Search for a message based on the search criteria described in *Search Parameters* on page 178.

   Depending on the search criteria you specify, the Search Result displays various parameters as described in *Search Results Parameters* on page 179.

Note: You must select the **Copy** function from the Select menu in the same row as the message that you want to copy.

3. Click **Copy** to display the **Copy Message** page. Enter the following details, as required, and click **Next**:

Field	Description
Message Name	The name of the copy of the SWIFT message. Required.
Sender ID	Identifier for the organization that is sending the SWIFT message. Optional. **Note:** The Sender ID value is required for some SWIFT messages since this value is used to perform validation against your data.
Receiver ID	Identifier for the organization receiving the SWIFT message. Optional. **Note:** The Receiver ID value is required for some SWIFT messages since this value is used to perform validation against your data.

Note: The status of a copied message is independent from the status of the source message. For example, if the source message was in Draft status when a copy of the message was made, the copied message continues to show the status as Draft even if the status of the source message has since changed to Sent or Rejected.

You are prompted with a Copy Message confirmation window with the following details:

Field	Description
Message Name	Displays the name of the copied SWIFT message.
Copy of Message	Displays the original SWIFT message name from which the copy was created.
Standard	Displays the SWIFT message standard.
Standard Version	Displays the version of the SWIFT standard.
Message Type	Displays the type of SWIFT message.
Sender ID	Identifier for the organization that is sending the SWIFT message. **Note:** If a Sender ID was not specified when you created the copy, this parameter contains the indicator Not Provided.
Receiver ID	Identifier for the organization receiving the SWIFT message. **Note:** If a Receiver ID was not specified when you created the copy, this parameter contains the indicator Not Provided.

4. Click **Finish** to create a copy of the SWIFT message. Once the copy is successfully created, the message "The system update completed successfully" is displayed.

5. Click **Return** to return to the main Message Entry Workstation page.

SWIFTNet Error Codes

Overview

The Translation service produces a Translation Status report. The report contains information about the translation of the document and any compliance errors. Errors from the input side of the map are listed under the Input tab. Errors from the output side of the map are listed under the Output tab.

Compliance errors can also be specified with the extended rule *cerror*.

This section describes all the translation status errors and error codes that may be generated when using SWIFTNet.

Text Validation Error Codes

The following table contains the error codes for SWIFTNet text validation:

SWIFTNet Error Code	Translator Report Error Number	Error Description
M50	901	Message length exceeded
M60	902	Non-SWIFTNet character encountered (a character not included in the <X>, <Y>, <Z> character sets. Also see error code T32.)
T01	904	Code word error. This check applies to: ◆ Field 39B in MT700,705,707,710,720,740,747. ◆ Field :22H:COAL subfield 3 in MT503,504.
T02	905	Unable to determine the reason for NAK. Please contact your CSC immediately for advice.
T03	906	Code word error. This check applies to: ◆ Field 26C, subfield 3, in MT600,601,604,605,606,607,608,609. ◆ Field 38B, subfield 1, in MT405. ◆ Field :22H:COLA subfield 3 in MT503,504,505,506,507.
T04	907	Code word error. This check applies to: ◆ Field 26C, subfield 4, in MT600,601,604,605,606,607,608,609. ◆ Field 38B, subfield 2, in MT405. ◆ Field :22H:COLA subfield 3 in MT503,504,505,506,507. ◆ Field :22H:REDE subfield 3 in MT528,529,536,537,548,575,578,584,586.

SWIFTNet Error Code	Translator Report Error Number	Error Description
T05	908	Code word error. This check applies to: ◆ Field 68B, subfield 4, in MT609. ◆ Field 38B, subfield 3, in MT405. ◆ Field :22H:DEPO subfield 3 in MT503,504,505,506.
T06	909	Code word error. This check applies to: ◆ Subfield 1, field 32F ◆ Subfield 5, field 68B,68C ◆ Subfield 3, field 60F,60M,62F,62M,64,65 in MT608 or in any message appended in common groups n92,n95,n96.
T07	910	Code word error. This check applies to: Subfield 3, field 33G. Field :22H:INOU subfield 3 in MT503,505,527,558.
T08	911	Code word error. This check applies to: ◆ Field 23, in MT102_STP,609. ◆ Field 26G, in MT550, or in any message appended in common groups n92,n95,n96. ◆ Field 71A in MT100,101,102,103_not_STP,103_STP,104,107,405,740, or in any message appended in common groups n92,n95,n96. ◆ Field :22H:REPR subfield 3 in MT527,558,569.
T09	912	Code word error. This check applies to: ◆ Subfield 3, field 23 in MT305,601. ◆ Subfield 3, field 68C in MT609, or in any message appended in common groups n92,n95,n96. ◆ Field :25D:COLL subfield 3 in MT507 when Data Source Scheme (DSS) is not present.
T10	913	The repetitive sequence occurred more than the maximum number of times permitted.
T11	914	The repetitive sequence occurred less than the minimum number of times required.
T12	915	Field, line, subfield, or component content error. Or, the format Reject/Return is not allowed for field 72 in this MT. Or, when "ISIN" is used at the beginning of line one in field 35B it must never be composed of lower-case letters, nor a mix of upper and lower-case letters. Or, in the ISITC MT521 or MT523 in field 35B the second line is mandatory and must not begin with any of the code words defined for the third and subsequent lines.

SWIFTNet Error Code	Translator Report Error Number	Error Description
T13	916	The field tag is not expected at this location in this MT. Either a mandatory field is missing, the sequence of fields is not correct, the specified field is not allowed at this point in the MT, or the specified field is not a defined SWIFTNet field (for example, the field tag is invalid), an end-of-text sequence (CRLF-) was encountered when it was not expected, or more than one end-of-text sequence occurs in this message.
		Or in a common group message (i.e. n92,n95,n96) within the list of "Copy of any field(s) of the original message", there are generic fields and non-generic fields that are not allowed to be mixed in the same message.
		Or in one of the following ISO15022 messages: MT502, 503, 504, 505, 506, 507, 508, 509, 513, 514, 515, 518, 524, 527, 528, 529, 530, 535, 536, 537, 538, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 558, 564, 565, 566, 567, 568, 569, 575, 576, 578, 584, 586, 587, 588, 589, an optional sequence of fields was used, however a field or field qualifier which is required within that sequence is missing, or field 16R is present but its related code word is inappropriate.
		Or in a qualifier table, for a particular generic field, in a "repeatable" order, there is a list of "OR" qualifiers but more than one qualifier has been used in the repetitions of the generic field in this sequence.

SWIFTNet Error Code	Translator Report Error Number	Error Description
T14	917	Subfield[N] (negative indicator) must not be used when the amount, number, or number count component is equal to zero. This check applies to:
		◆ Field 19A in MT502,513-515,518,527-529,535-537,540-548,558,564,566-567, 569,575,578,584,586-588.
		◆ Field 32H in MT320,330,362,571.
		◆ Field 32N in MT646.
		◆ Field 33N in MT646,
		◆ Field 34E in MT320,330,341,571.
		◆ Field 34N in MT646
		◆ Field 35H in MT571,581.
		◆ Field 37G in MT320,330,362,644.
		◆ Field 37M in MT340,341,362,644.
		◆ Field 37R in MT341,360,361,362,644.
		◆ Fields 60A,62A,62B in MT572.
		◆ Field 60B in MT571,572.
		◆ Field 92A in MT502,506,513-515,518,527-529,540-547,558,564-566,568-569, 576,578,584,586-588.
		◆ Field 92E in MT564,566.
		◆ Field 93B in MT535,536,564-566,568.
		◆ Field 93C in MT535,564-566,568.
		◆ Field 93D in MT575.
		◆ Field 98D in MTs 564, 566.
		◆ Field 98E in MTs 513-515, 518, 528, 529, 540-548
		◆ 578, 586.Field 99A in MT506,513-515,518,528-529,535-536,540-547,569,575,578,586.
		◆ Any of the above fields in common groups n92,n95,n96.
T15	918	Sign is not valid. <SIGN> must be either '+' or '-'.
T16	919	Time offset is not valid. <OFFSET> has the same format as time <HHMM>.
T17	920	Field, line, subfield, or component consists of blanks, <CRLF>, or it is missing a mandatory line, subfield, or component.

SWIFTNet Error Code	Translator Report Error Number	Error Description
T18	921	Component is not in the format 3!n and/or it is not within the range 100-999. This check applies to: ◆ Field 11R,11S: the first component must have the format 3!n and must be within the range 100-999. ◆ Field 12, and MT105,106: this component must have the format 3!n and must be within the range 100-999. ◆ Field 12, and MT NOT=(M105,106): this component must have the format 3!n. ◆ Refer to T88 for additional special exception checking. ◆ Field 61: if the first character of subfield 6 is 'S' then the next three characters must have the format 3!n and must be within the range 100-999.
T19	922	Code word error. This check applies to: ◆ Subfield 1 of field 87E,87F. ◆ Field :25D:4!c//<Status> subfield 3 (Status) in MT507 when Data Source Scheme (DSS) is not present. ◆ Field :22H:4!c//<Indicator> subfield 3 (Indicator) in MT 307,321.
T20	923	Code word error in subfield 1, component 3, of field 32K or 33K.
T21	924	(Available).
T22	925	A common reference mismatch exists between field 22, subfield 2, component 2 and one of the following: ◆ Field 36 in MT305. ◆ Field 33G subfield 2 in MT600. ◆ Field 32B subfield 2 in MT601. Or, a common reference mismatch exists between field 22C, component 2, and one of the following: ◆ Field 30P, YYMM of YYYYMMDD in MT360-362,364-365. ◆ Field 36 sequence B in MT300. ◆ Field 36 sequence B in MT303. ◆ Field 36A sequence C in MT303. ◆ Field 36 sequence D or field 37U sequence G in MT306. ◆ Field 37G sequence B in MT320,330. ◆ Field 37M sequence B in MT340,341. ◆ Field 37J sequence B in MT350.
T23	926	Subfield 8 in field 61, subfield 5 in field 66A, or subfield 6 in field 26C is too long or contains only '//', or subfield 2 in fields 26A or 26B is too long or contains only '/'.

SWIFTNet Error Code	Translator Report Error Number	Error Description
T24	927	Subfield 7 in field 61, subfield 4 in field 66A, subfield 5 in field 26C, subfield 1 in fields 26A or 26B is missing or is too long.
T25	928	Subfields 7 or 8 in field 61, subfield 4 or 5 in field 66A, subfield 5 or 6 in field 26C, subfield 1 or 2 in fields 26A or 26B has improper content.
T26	929	Subfield 7 or 8 in field 61, subfield 4 or 5 in field 66A, subfield 5 or 6 in field 26C, subfield 1 or 2 in fields 26A or 26B has improper content. This check applies to: ◆ fields 20,20C,21,21A,21F,21G,21P,21R.
T27	930	BIC incorrectly formatted or invalid.
T28	931	SWIFTNet BIC is not a valid destination.
T29	932	SWIFTNet BIC contains an invalid branch code.
T30	933	Excessive lines, subfields, or components were found in this field.
T31	934	The line, subfield, or component separator or delimiter is missing or incorrect.
T32	935	An expected subfield, component, or component separator was not found.
T33	936	The length of the field, line, subfield, or component contents is too long, or, the component consists of one or more hidden characters, or, the component consists of one or more imbedded characters which are inconsistent with the defined field format, or the characters do not belong to the correct character set.
T34	937	The length of the field, line, subfield, or component contents is too short.
T35	938	Code word error in subfield 1, field 26C.

SWIFTNet Error Code	Translator Report Error Number	Error Description
T36	939	Code word error. This check applies to: ◆ Field 14A in MT360,361. ◆ Field 14D in MT320,330,340,350,360,361. ◆ Field 14J in MT360,361. ◆ Field 17A in MT206,300,303,306,360,361,405. ◆ Field 17B in MT500-505,513-515,518-519,527-529,535-538,540,547,558,564, 569,574W8BENO,574IRSLST,575-578,584,586-588. ◆ Field 17F in MT304,306,340,360,361,405. ◆ Field 17G,17N,17O in MT304. ◆ Field 17T,17U in MT300. ◆ Field 22A in MT293,300,303,304,306,320,330,340,341,350,360,361,362,364,365. ◆ Field 23B in MT 103_not_STP,103_STP,303. ◆ Field 94A in MT 300, 303, 304, 305, 306, 320, 330, 340, 341, 350, 360, 361, 362, 364, 365, 600, 601.
T37	940	Code word error. This check applies to: ◆ Subfield 2, field 35H when used with MT581 and with any message appended in common groups n92,n95,n96. ◆ Subfield 2, field 35T when used with MT552 and with any message appended in common groups n92,n95,n96.
T38	941	Illogical time specified.
T39	942	Code word error in subfield 2, field 66A.
T40	943	Missing amount/number or incorrect amount/number first character.
T41	944	Code word error in subfield 3, field 66A.
T42	945	Code word error in subfield 3, field 35U.
T43	946	The decimal separator in the amount/number subfield or component is missing, is not a valid character, or more than one separator is present.
T44	947	The SWIFTNet BIC exists but it is not enabled for FIN, or it is not a cutover.
T45	948	Invalid non-SWIFTNet BIC.
T46	949	A Test-and-Training destination must not be used in a LIVE message.

SWIFTNet Error Code	Translator Report Error Number	Error Description
T47	950	In an ERI field the data part of a code word /OCMT/, or /CHGS/ was being validated but the ending separator (such as, the third '/') is missing. This error is also a code word error that applies to:

- Field 14G, subfield 1, in MT360-361.

- Field 22, subfield 1, in MT305.

- Field 23A, subfield 1, in MT360-362,364-365.

- Field 23E. subfield 1, in MT101,103_not_STP,104,107,206,207,256,405,416.

- Field 35B, in ISITC MT521,523: invalid or duplicated code word in line 3 or following.

- Field 39P, subfield 1, in MT303.

- Field 57D, line 1, in ISITC MT521,523.

- Field 61, subfield 9, duplicated code words: /OCMT/ or /CHGS/.

- Field 72 (narrative), duplicated code words: /OCMT/ or /CHGS/.

- Field 72 (structured), duplicated code words: /OCMT/, /CHGS/, or /INS/.

- Field 77A, duplicated code words: /OCMT/ or /CHGS/.

- Field 72, in ISITC MT521,523: invalid or duplicated code word, or in ISITC MT523 mandatory code word is missing.

- Field 77D, line 1, sequence C in MT303.

- Field 77D, lines 1-6, in ISITC MT521,523: invalid or duplicated code word.

- Field 77H, subfield 1, in MT306,340,360,361.

- Field 79, duplicated code words: /OCMT/ or /CHGS/.

- Field 85D, line 1, in ISITC MT521.

- Field 86, duplicated code words: /OCMT/ or /CHGS/.

- Field 87D, line 1, in ISITC MT521,523.

- Field 88D, line 1, in ISITC MT521,523.

SWIFTNet Error Code	Translator Report Error Number	Error Description
T48	951	Code word error. This check applies to: ◆ Field 14G, subfield 2, in MT360-361. ◆ Field 22K, subfield 1, in MT306. ◆ Field 23A, subfield 2, in MT360-362,364-365. ◆ Field 23D, in MT340-341. ◆ Field 23E, subfield 1, in MT103_STP. ◆ Field 31P, subfield 2, in ISITC MT521,523. ◆ Field 35B, in ISITC MT521,523: invalid code word in line 1. ◆ Field 38E, subfield 2, MT360-361. ◆ Field 38G, subfield 2 or 4, MT340,360-361. ◆ Field 38H, subfield 2 or 4, MT360-361.
T49	952	(Available).
T50	953	Date error, or the value of "year" (YY) in a Value Date component <DATE2> is invalid.
T51	954	Code word 'C','D','RC','RD','EC','ED' error.
T52	955	Invalid currency code or price code 'PCT','REN', or 'YLD'.
T53	956	Code word error in subfield 6, component 1 of field 61.
T54	957	The format of the first line of Field 50F (Party Identifier) is invalid. This check applies to: ◆ Field 50F in MTs 101, 102, 102_STP, 103, 103_STP, 210, 910. ◆ This check applies to all MTs containing field 50F, when appended to Common Group MTs n92, n95, n96.
T55	958	Code word error. This check applies to Subfield 1, component 1: ◆ Field 50F in MTs 101, 102, 102_STP, 103, 103_STP, 210, 910. ◆ This check applies to all MTs containing field 50F, when appended to Common Group MTs n92, n95, n96.
T56	959	Code word error. This check applies to Subfield 2, component 1: ◆ Field 50F in MTs 101, 102, 102_STP, 103, 103_STP, 210, 910. ◆ This check applies to all MTs containing field 50F, when appended to Common Group MTs n92, n95, n96.
T57	960	Code word error in subfield 2 of fields 31H, 31J, or 31X.
T58	961	Code word error. This check applies to: ◆ Subfield 1 of field 35A,35N,35P,35S. ◆ Subfield 2 of field 35H,35T. ◆ Subfield 1 in the 2nd occurrence of field 35A in MT550.

SWIFTNet Error Code	Translator Report Error Number	Error Description
T59	962	(Available).
T60	963	Code word error. This check applies to: ◆ Field 26F in MT306. ◆ Field 40A in MT700,705.
T61	964	Code word 'D' or 'M' error. This check applies to: ◆ Field 32K, subfield 1, in MT400,405,410,416,420,422,430. ◆ Field 33k, subfield 1, in MT430. ◆ Field 37(A-F), subfield 2, in MT516,644,645,646. ◆ Field 38J, subfield 1, in MT320,330.
T62	965	Either the first subfield <DATE2>[<HHMM>] or the second subfield 7!a but not both must be present. If optional subfield 1 is used, component 1 <DATE2> of this subfield must be present. This check applies to: Fields 31H,31J,31X.
T63	966	Error in component 2 of field 22C or subfield 2, component 2, of field 22. When the last character of this component is zero '0' and the preceding character is not one '1' then the entire component must consist of zeros.
T64	967	Code word error. This check applies to: ◆ Field 12F in MT306. ◆ Field 40B, line 1, in MT710,720.
T65	968	(Available).
T66	969	Code word error. This check applies to: ◆ Field 12E in MT306. ◆ Field 40B, line 2, in MT710,720.
T67	970	Code word error. This check applies to: ◆ Field 17R in MT320,330. ◆ Field 17V in MT306. ◆ Field 49 in MT700,710,720.
T68	971	Code word error. This check applies to: ◆ Field 24D, subfield 1, in MT300,306,320,330,240,360,361. ◆ Field 41A, subfield 2, in MT700,705,710,720,740. ◆ Field 41D, subfield 2, in MT700,705,710,720,740.
T69	972	(Available).
T70	973	Either the Account number, the Place, or both must be present.

SWIFTNet Error Code	Translator Report Error Number	Error Description
T71	974	Code word error. This check applies to: ◆ Field 22D in MT360,361,364,365,405. ◆ Field 22E in MT405. ◆ Field 22G in MT306.
T72	975	Code word error. This check applies to: Field 22J in MT306. Field 23C, subfield 1, in MT405. Field 23F, subfield 1, in MT405.
T73	976	Invalid country code. Please refer to the BIC Directory General Information -Country Codes-.
T74	977	The currency code must be the same for each indicated subfield in the field.
T75	978	In MT405,n92,n95,n96: Field 38B. When subfields 1 and 2 contain "MONT/OTHR" then subfield 3 is mandatory, otherwise subfield 3 is not allowed.
T76	979	The first character in the first line of this field must be a '/', and there must be at least another line, but not more than 5 lines. This check applies to field 50H.
T77	980	If the first character of the first line of this component or sub-component is a '/', then there must be at least another line, but not more than 5 lines. Otherwise, no more than 4 lines are allowed. This check applies to: Fields 42D,50K,(50-58)D,59,(82-88)D, and subfield 2 of field 87F.

SWIFTNet Error Code	Translator Report Error Number	Error Description
T78	981	Invalid or duplicated code word, or a mandatory code word is missing. This check applies to: ◆ Field 53J, subfield 1 of each line, in MTs 300, 303 and 304, 306, 320, 330, 340, 341, 350. ◆ Field 56J, subfield 1 of each line, in MTs 300, 303 and 304, 306, 320, 330, 340, 341, 350. ◆ Field 57J, subfield 1 of each line, in MTs 300, 303 and 304, 306, 320, 330, 340, 341, 350. ◆ Field 58J, subfield 1 of each line, in MTs 300, 303 and 304, 306, 320, 330, 340, 341, 350. ◆ Field 82J, subfield 1 of each line, in MTs 300, 303 and 304, 305, 306, 320, 330, 350, 600, 601. ◆ Field 83J, subfield 1 of each line, in MTs 300, 303 and 304, 305, 306, 320, 330, 350, 600, 601. ◆ Field 84J, subfield 1 of each line, in MTs 300, 306, 320, 330. ◆ Field 85J, subfield 1 of each line, in MTs 300, 320, 330. ◆ Field 86J, subfield 1 of each line, in MTs 306, 320, 330, 340, 341, 350. ◆ Field 87J, subfield 1 of each line, in MTs 300, 303 and 304, 305, 306, 320, 330, 350, 600, 601. ◆ Field 88J, subfield 1 of each line, in MTs 300, 320.
T79	982	(Available).
T80	983	<Field 72 Reject/Return> or <Field 79 Reject Return> code word error, or mandatory code word missing, or code word not in proper sequence.
T81	984	Format of <Field 72 Reject/Return> is not allowed in this message. This check applies to: Field 72, MT102_STP,103_STP.
T82	985	ERI format is not allowed in this message. This check applies to: Field 72, MT102_STP,103_STP.
T83	986	(Available).
T84	987	(Available).
T85	988	Code word error. This check applies to Field 23G, subfield 2, in MTs 307, 308, 321, 380, 381, 500-510, 513-515, 517-519, 524, 527-529, 530, 535-538, 540-549, 558, 564-569, 574IRSLST, 574W8BENO, 575, 576, 578, 584, 586-589, or in any message appended in common groups n92, n95 or n96.

SWIFTNet Error Code	Translator Report Error Number	Error Description
T86	989	Code word error. This check applies to: Field 23G, subfield 1, in MTs 307, 308, 321, 380, 381, 500-510, 513-515, 517-519, 524, 527-529, 530, 535-538, 540-549, 558, 564-569, 574IRSLST, 574W8BENO, 575, 576, 578, 584, 586-589, or in any message appended in common groups n92, n95 or n96.
T87	990	In a generic field the colon ':' delimiter is not present at the expected position.
T88	991	Field 12 in MT570 may only consist of 571,572,573,577. Field 12 in MT920 may only consist of 940,941,942,950. Field 12 in MT973 may only consist of 971,972,998. Field 13A in MT507 may only consist of 503,504,505. Field 13A in MT549 may only consist of 509,535-538,548,567,575-577,584, 586,589.
T89	992	In a generic field either the qualifier is invalid, the qualifier is duplicated, a mandatory qualifier is missing, or the qualifier format is not valid.
T90	993	In a generic field either the issuer code format is invalid, the mandatory issuer code is missing, or the generic field format is invalid.
T91	994	In a generic field the slash '/' delimiter is not present at the expected position.
T92	995	Code word error. This check applies to: ◆ Field 16R in ISO15022. ◆ Field 16S in ISO15022.
T93	996	Code word error. This check applies to: Field 22B in MT320,330,364,365.
T94	997	In field 22, subfield 2, or in field 22C, the components 1 and 3, the values '0' and '1' are not permitted in <LC1> and <LC2>. However, the value '0' is allowed in the second (rightmost) position if the sender or receiver is a test and training user.
T95	998	In field 22, subfield 2, or in field 22C, the components 1 and 3 do not contain the bank code and location code of the message sender, and/or the bank code and location code of the message receiver.
T96	999	In field 22, subfield 2, or in field 22C, the components 1 and 3 are not in alphabetical sequence.
T97	1000	Code word error. This check applies to: Field 28E, subfield 2, in MT206,506,535-538,569,574WBENO,574IRSLST,575, 576,584,586.
T98	1001	(Available).

SWIFTNet Error Code	Translator Report Error Number	Error Description
T99	1002	A special function has been declared in the validation syntax that is not recognized. **Note:** If you receive this error, it indicates that a special function was declared in a syntax that was not recognized by the translator.

Specific Error Codes for MUG-textval Rules

The following table contains the specific error codes for SWIFTNet MUG-textval rules:

Note: If the Translator Report Error Number column is blank, this indicates that there is not currently a translator code equivalent for the SWIFTNet Error Code. The translator report codes are used to generate errors from within extended rules using the cerror() function. If you want to validate the conditions corresponding to G18 through G24 in an extended rule, we recommend that you use the translator report code for a similar error (G01 (1003) through G17 (1019)).

SWIFTNet Error Code	Translator Report Error Number	Error Description
G01	1003	AU/PDS: in MT100,103,202, the letter option of the "selected field" is not A or D.
G02	1004	AU/PDS: in MT100,103,202, the format of the "selected field" option A is not valid.
G03	1005	AU/PDS: in MT100,103,202, the format of the "selected field" option D is not valid.
G04	1006	AU/PDS: the "selected field" is missing. At least one of the following fields must be present: MT100,103: fields 56a, 57a.
G05	1007	LVTS: if 2 LVTS members and the first 6 characters of their destination ID are different, exchange a SWIFT message type 100,103,205, and if the currency code used in tag 32A is "CAD" then the tag 103 must be present in the User Header and it must contain the code "CAD".
G06	1008	REMIT: in a SWIFT message MT103, the field 77T and the tag 119 with the code word "REMIT" (in the User Header) must either both be present or absent.
G07	1009	CLS: in an MT300 eligible for the FIN-Copy service CLS or CLT, any field 53 present in sequence B must be used with the letter option 'A'.
G08	1010	CLS: in an MT300 eligible for the FIN-Copy service CLS or CLT, both fields 57 in sequences B1 and B2 (index 20,24) must be used with the letter option 'A', field 57a: of subsequence B1 must contain the "CLSB" BIC bank code.
G09	1011	CLS: in an MT300 eligible for the FIN-Copy service CLS or CLT, if the tag 17U is used it must contain the value "N".

SWIFTNet Error Code	Translator Report Error Number	Error Description
G10	1012	CLS: in an MT300 eligible for the FIN-Copy service CLS or CLT, any field 56 present in sequence B must be used with the letter option 'A'.
G11	1013	CLS: in an MT300 eligible for the FIN-Copy service CLS or CLT, if field 82 is present in sequence A it must be used with letter option 'A'.
G12	1014	CLS: in an MT300 eligible for the FIN-Copy service CLS or CLT, if field 87 is present in sequence A it must be used with letter option 'A'.
G13	1015	CLS: in an MT304 sent to the CLSB server, field 94A must contain the code "ASET".
G14	1016	CLS: in an MT304 sent to the CLSB server, field 82 in sequence A must be used with letter option 'A'.
G15	1017	CLS: in an MT304 sent to the CLSB server, field 87 in sequence A must be used with letter option 'A'.
G16	1018	CLS: in an MT304 sent to the CLSB server, any field 53 present in sequence B must be used with the letter option 'A'.
G17	1019	CLS: in an MT304 sent to the CLSB server, both fields 57 in sequence B must be used with letter option 'A' and must contain the ""CLSB"" BIC bank code. **Note:** Field 57A, index 19, is mandatory.
G18		In MT 305 eligible for the FIN-Copy service CLS or CLT, field 53a must be used with option A.
G19		In MT 305 eligible for the FIN-Copy service CLS or CLT, field 56a must be used with option A.
G20		In MT 305 eligible for the FIN-Copy service CLS or CLT, field 57A must be present.
G21		In MT 305 eligible for the FIN-Copy service CLS or CLT, field 57A must be present.
G22		In MT 305 eligible for the FIN-Copy service CLS (or CLT,) when the emitter and receiver are both CLS (or CLT) members, then field 57A must contain CLSB****.
G23		In MT 305 eligible for the FIN-Copy service CLS (or CLT), when the emitter is CLS (or CLT) member and receiver is not, and field 34R is present, then field 57A must contain CLSB****. When the emitter is CLS (or CLT) member and receiver is not, and field 34P is present, then field 56A must contain CLSB****.
G24		In MT 305 eligible for the FIN-Copy service CLS (or CLT), both fields 56A and 57A must not contain the CLSB BIC bank code at the same time.

Special Error Codes for Value-Added Service Messages

The following table contains the special error codes for SWIFTNet value-added service messages:

SWIFTNet Error Code	Translator Report Error Number	Error Description
B01	1020	PAC Trailer used for non-Premium service message. Message has PAC trailer but sender or receiver, or both, are not members of Premium service.
B02	1021	(Available)
B03	1022	103:LCH present in the message, but sender, receiver, or both are not members of LCH, or the message type is not allowed for LCH, or, '103:TPS' present in the message, but sender, receiver, or both are not members of TPS, or the message type is not allowed for TPS.
B04	1023	(Available)
B05	1024	A system error has occurred. The user should contact their local Customer Service Center for further information.

Message Syntax and Semantic Rule Codes

The following section contains the error codes for SWIFTNet message syntax and semantic rules.

The C, D, and E error code ranges refer to rule numbers (these represent the message validation rules), are documented as follows:

✦ Rules 000-099 as C00-C99

✦ Rules 100-199 as D00-D99

✦ Rules 200-299 as E00-E99

Rules 000-099

SWIFTNet Error Code	Translator Report Error Number
C00	1025
C01	1026
C02	1027
C03	1028

SWIFTNet Error Code	Translator Report Error Number
C04	1029
C05	1030
C06	1031
C07	1032
C08	1033
C09	1034
C10	1035
C11	1036
C12	1037
C13	1038
C14	1039
C15	1040
C16	1041
C17	1042
C18	1043
C19	1044
C20	1045
C21	1046
C22	1047
C23	1048
C24	1049
C25	1050
C26	1051
C27	1052
C28	1053
C29	1054
C30	1055
C31	1056
C32	1057
C33	1058
C34	1059

SWIFTNet Error Code	Translator Report Error Number
C35	1060
C36	1061
C37	1062
C38	1063
C39	1064
C40	1065
C41	1066
C42	1067
C43	1068
C44	1069
C45	1070
C46	1071
C47	1072
C48	1073
C49	1074
C50	1075
C51	1076
C52	1077
C53	1078
C54	1079
C55	1080
C56	1081
C57	1082
C58	1083
C59	1084
C60	1085
C61	1086
C62	1087
C63	1088
C64	1089
C65	1090

SWIFTNet Error Code	Translator Report Error Number
C66	1091
C67	1092
C68	1093
C69	1094
C70	1095
C71	1096
C72	1097
C73	1098
C74	1099
C75	1100
C76	1101
C77	1102
C78	1103
C79	1104
C80	1105
C81	1106
C82	1107
C83	1108
C84	1109
C85	1110
C86	1111
C87	1112
C88	1113
C89	1114
C90	1115
C91	1116
C92	1117
C93	1118
C94	1119
C95	1120
C96	1121

SWIFTNet Error Code	Translator Report Error Number
C97	1122
C98	1123
C99	1124

Rules 100-199

SWIFTNet Error Code	Translator Report Error Number
D00	1125
D01	1126
D02	1127
D03	1128
D04	1129
D05	1130
D06	1131
D07	1132
D08	1133
D09	1134
D10	1135
D11	1136
D12	1137
D13	1138
D14	1139
D15	1140
D16	1141
D17	1142
D18	1143
D19	1144
D20	1145
D21	1146
D22	1147
D23	1148
D24	1149

SWIFTNet Error Code	Translator Report Error Number
D25	1150
D26	1151
D27	1152
D28	1153
D29	1154
D30	1155
D31	1156
D32	1157
D33	1158
D34	1159
D35	1160
D36	1161
D37	1162
D38	1163
D39	1164
D40	1165
D41	1166
D42	1167
D43	1168
D44	1169
D45	1170
D46	1171
D47	1172
D48	1173
D49	1174
D50	1175
D51	1176
D52	1177
D53	1178
D54	1179
D55	1180

SWIFTNet Error Code	Translator Report Error Number
D56	1181
D57	1182
D58	1183
D59	1184
D60	1185
D61	1186
D62	1187
D63	1188
D64	1189
D65	1190
D66	1191
D67	1192
D68	1193
D69	1194
D70	1195
D71	1196
D72	1197
D73	1198
D74	1199
D75	1200
D76	1201
D77	1202
D78	1203
D79	1204
D80	1205
D81	1206
D82	1207
D83	1208
D84	1209
D85	1210
D86	1211

SWIFTNet Error Code	Translator Report Error Number
D87	1212
D88	1213
D89	1214
D90	1215
D91	1216
D92	1217
D93	1218
D94	1219
D95	1220
D96	1221
D97	1222
D98	1223
D99	1224

Rules 200-299

SWIFTNet Error Code	Translator Report Error Number
E00	1225
E01	1226
E02	1227
E03	1228
E04	1229
E05	1230
E06	1231
E07	1232
E08	1233
E09	1234
E10	1235
E11	1236
E12	1237
E13	1238
E14	1239

SWIFTNet Error Code	Translator Report Error Number
E15	1240
E16	1241
E17	1242
E18	1243
E19	1244
E20	1245
E21	1246
E22	1247
E23	1248
E24	1249
E25	1250
E26	1251
E27	1252
E28	1253
E29	1254
E30	1255
E31	1256
E32	1257
E33	1258
E34	1259
E35	1260
E36	1261
E37	1262
E38	1263
E39	1264
E40	1265
E41	1266
E42	1267
E43	1268
E44	1269
E45	1270

SWIFTNet Error Code	Translator Report Error Number
E46	1271
E47	1272
E48	1273
E49	1274
E50	1275
E51	1276
E52	1277
E53	1278
E54	1279
E55	1280
E56	1281
E57	1282
E58	1283
E59	1284
E60	1285
E61	1286
E62	1287
E63	1288
E64	1289
E65	1290
E66	1291
E67	1292
E68	1293
E69	1294
E70	1295
E71	1296
E72	1297
E73	1298
E74	1299
E75	1300
E76	1301

SWIFTNet Error Code	Translator Report Error Number
E77	1302
E78	1303
E79	1304
E80	1305
E81	1306
E82	1307
E83	1308
E84	1309
E85	1310
E86	1311
E87	1312
E88	1313
E89	1314
E90	1315
E91	1316
E92	1317
E93	1318
E94	1319
E95	1320
E96	1321
E97	1322
E98	1323
E99	1324

Knn: Code Word Validation in Generic Fields

The two digits **nn** indicate the field ID.

SWIFTNet Error Code	Translator Report Error Number
K00	1325
K01	1326
K02	1327

SWIFTNet Error Code	Translator Report Error Number
K03	1328
K04	1329
K05	1330
K06	1331
K07	1332
K08	1333
K09	1334
K10	1335
K11	1336
K12	1337
K13	1338
K14	1339
K15	1340
K16	1341
K17	1342
K18	1343
K19	1344
K20	1345
K21	1346
K22	1347
K23	1348
K24	1349
K25	1350
K26	1351
K27	1352
K28	1353
K29	1354
K30	1355
K31	1356
K32	1357
K33	1358

SWIFTNet Error Code	Translator Report Error Number
K34	1359
K35	1360
K36	1361
K37	1362
K38	1363
K39	1364
K40	1365
K41	1366
K42	1367
K43	1368
K44	1369
K45	1370
K46	1371
K47	1372
K48	1373
K49	1374
K50	1375
K51	1376
K52	1377
K53	1378
K54	1379
K55	1380
K56	1381
K57	1382
K58	1383
K59	1384
K60	1385
K61	1386
K62	1387
K63	1388
K64	1389

SWIFTNet Error Code	Translator Report Error Number
K65	1390
K66	1391
K67	1392
K68	1393
K69	1394
K70	1395
K71	1396
K72	1397
K73	1398
K74	1399
K75	1400
K76	1401
K77	1402
K78	1403
K79	1404
K80	1405
K81	1406
K82	1407
K83	1408
K84	1409
K85	1410
K86	1411
K87	1412
K88	1413
K89	1414
K90	1415
K91	1416
K92	1417
K93	1418
K94	1419
K95	1420

SWIFTNet Error Code	Translator Report Error Number
K96	1421
K97	1422
K98	1423
K99	1424

Header Validation Error Codes

The following table contains the specific H and U error codes for SWIFTNet header validations that Application supports:

SWIFTNet Error Code	Translator Error Code or Description of How This Validation is Handled and Enforced	Error Description
H01	Translator Error Code 1425—enforced by the envelope map	Basic Header no present or format error block 1
H02	Translator Error Code 1426 (inbound)—Application generates this error	Application Identifier not 'A' (GPA) or 'F' (FIN)
H03	Translator Error Code 1427(inbound and outbound)—Application generates this error	Invalid Service Message identifier (must be 01 or 21)
H4-H9	Not used	Available
H10	Translator Error Code 1434—not generated or enforced by Application	Bad LT address or application not enabled for the LT
H15	Translator Error Code 1439—not generated or enforced by Application	Bad session number
H20	Translator Error Code 1444—not generated or enforced by Application	Error in the ISN
H21	Translator Error Code 1445 (outbound)—Application generates this error	Error in the message sender's branch code
H25	Translator Error Code 1449—enforced by the envelope map	Application header format error or not present when mandatory
H26	Translator Error Code 1450—enforced by the envelope map	Input/output identifier not "I" (on input from LT)
H30	Translator Error Code 1454 (inbound and outbound)—Application generates this error	Message type for a SWIFT message not found in code list.
H40	Translator Error Code 1464 (inbound and outbound)—Application generates this error	Message priority other than S (System) for a message type < 100, or message priority of S for message type < 100

SWIFTNet Error Code	Translator Error Code or Description of How This Validation is Handled and Enforced	Error Description
H50	Translator Error Code 1474 (inbound and outbound)—Application generates this error	Receiver ID (ID + 'X' + branch code) not found in code list.
H51	Translator Error Code 1475 (outbound)—Application generates this error	Message Type < 100 must have a receiver ID of SWFTXXX and a branch code of XXX.
H52	Translator Error Code 1476—not generated or enforced by Application	MT 072, selection of Test and Training mode/version, MT 077 Additional Selection Criteria for SWIFT are not allowed while a SWIFT session is open
H55	Translator Error Code 1479—not generated or enforced by Application	Message type not allowed for fallback session for MT 030
H80	Translator Error Code 1504—enforced by the envelope map	Delivery option error
H81	Translator Error Code 1505—enforced by the envelope map	Obsolescence period error
H98	Translator Error Code 1522 (inbound and outbound)—Application generates this error	Sender ID (ID + 'X' + branch code) not found in code list.
H99	Translator Error Code 1523—enforced by the envelope map	Error can be one of the following: ◆ Invalid receiver destination (invalid character or LT identification is not "X") ◆ Invalid date or time (not numeric or not within range)
U00	Translator Error Code 1524—enforced by the envelope map	Bad block 3 format
U01	Translator Error Code 1525—enforced by the envelope map	Bad bank priority
U02	Translator Error Code 1526—enforced by the envelope map	Bad MUR
U03	Translator Error Code 1527—not generated or enforced by Application	Neither bank priority nor MUR present
U07	Translator Error Code 1531 (outbound)—Application generates this error	User Header not permitted for user-to-system messages (that is, message type < 100)

SWIFTNet Error Code	Translator Error Code or Description of How This Validation is Handled and Enforced	Error Description
U08	Translation Error Code 1532 (inbound and outbound)—Application generates this error	Tag 119 is not one of the following: ◆ REMIT (any message) ◆ RFDD (any message) ◆ STP (102 or 103) ◆ COMM (503, 504, 505, 506, or 507) ◆ CRPR (503, 504, 505, 506, or 507) ◆ CRSP (503, 504, 505, 506, or 507) ◆ CRTL (503, 504, 505, 506, or 507) ◆ EXTD (503, 504, 505, 506, or 507) ◆ FIXI (503, 504, 505, 506, or 507) ◆ FORX (503, 504, 505, 506, or 507) ◆ LIQU (503, 504, 505, 506, or 507) ◆ OTCD (503, 504, 505, 506, or 507) ◆ PAYM (503, 504, 505, 506, or 507) ◆ REPO (503, 504, 505, 506, or 507) ◆ SBSB (503, 504, 505, 506, or 507) ◆ SCRP (503, 504, 505, 506, or 507) ◆ SECL (503, 504, 505, 506, or 507) ◆ SLEB (503, 504, 505, 506, or 507) ◆ TCRP (503, 504, 505, 506, or 507) ◆ W8BENO (574) ◆ IRSLST (574)
U09	Translation Error Code 1533 (inbound and outbound)—Application generates this error	Tag 119 present for a message type other than 102, 103, 104, 503, 504, 505, 506, 507, 521, 523, and 574.

Error Codes for SWIFT MX Messages

The following are the error codes the translator uses for SWIFT MX messages and their corresponding SWIFTNet error code:

Translator Error Code	SWIFT Error Code
1633	AccruedInterestAmountSignRule
1634	BalanceAtSafekeepingPlaceRule
1635	BalanceForAccountOrSubAccountDetailsRule
1636	Party1Rule
1637	StreetNameAndOrPostOfficeBoxRule
1638	ValueRule
1639	AccumulationPeriodRule
1640	AddressRule
1641	AmountAndOrRateRule
1642	AmountSignRule
1643	BalanceForSubAccountRule
1644	BeneficiaryRule
1645	BulkCashSettlementDetails1Rule
1646	BulkCashSettlementDetails2Rule
1647	BulkCashSettlementDetails3Rule
1648	BulkCashSettlementDetails4Rule
1649	BulkCashSettlementDetails5Rule
1650	BulkCashSettlementDetails6Rule
1651	CalculationBasisRule
1652	CashSettlementDateRule
1653	ClientReferenceRule
1654	CommercialAgreementRule
1655	CorporateRule
1656	CurrencyPredefinedPeriodsRule
1657	CurrencyPriceChangeRule
1658	CurrencyUserDefinedPeriodsRule
1659	CurrentYearRule
1660	CurrentYearSubscriptionDetailsRule

Translator Error Code	SWIFT Error Code
1661	CurrentYearTypeRule
1662	DateOrDateCodeRule
1663	DeliverersCustodianDetailsRule
1664	DeliverersCustodianRule
1665	DeliverersIntermediaryDetailsRule
1666	DesignationRule
1667	ExceptionalCashFlowIndicatorRule
1668	ExchangeConversionRule
1669	MultipleSwitchExecutionRule
1770	ExtendedPartyRole2Rule
1771	ExtendedPartyRoleRule
1772	FinancialInstrumentQuantity1Rule
1773	FloorAmountRule
1774	FutureSettlementDateRule
1775	GoodTIllOrderRule
1776	InvestmentAccountDetailsRule
1777	InvestmentAccountIdentificationRule
1778	InvestmentAccountRule
1779	InvestorRule
1780	MailingIndicatorRule
1781	MessageNameAndReferenceGuideline
1782	MessageNameRule
1783	NameAndAddress1Rule
1784	NetAmountGrossAmountRule
1785	OrderOriginatorEligibility1Rule
1786	OrderOriginatorEligibility3Rule
1787	OrderOriginatorEligibility4Rule
1788	OrderPriceAndForeignExchangeRule
1789	OrderPriceAndLimitOrderRule
1790	OtherCodeRule
1791	OtherReferenceRule

Translator Error Code	SWIFT Error Code
1792	OtherStatusRule
1793	PEPISARule
1794	PendingAdditionalInformation
1795	PhysicalDeliveryDetailsRule
1796	PhysicalTransferDetailsRule
1797	PorfolioRule
1798	PreviousOrOtherReferenceOrAccountApplicationIdentificationRule
1799	PreviousReferenceRule
1800	PriceDetailsRule
1801	ReceiversCustodianDetailsRule
1802	ReceiversCustodianRule
1803	ReceiversIntermediaryDetailsRule
1804	ReferenceRule
1805	ReferredPlacementAgentRule
1806	RegistrationAddressIndicator1Rule
1807	RegistrationAddressIndicator2Rule
1808	RelatedPartiesDetailsRule
1809	RelatedReferenceRule
1810	RequestedSettlementCurrencyRule
1811	RoundingDirectionAndModulusRule
1812	RoundingMethodRule
1813	RoundingRule
1814	SettlementCurrency1Rule
1815	SettlementCurrency2Rule
1816	SettlementCurrency3Rule
1817	SettlementCurrency4Rule
1818	SettlementCurrency5Rule
1819	StatisticsCurrency1Rule
1820	StatisticsCurrency2Rule
1821	StopPriceAndStopOrderRule
1822	TaxRule

Translator Error Code	SWIFT Error Code
1823	TotalAmountYearToDateRule
1824	TotalNumberRule
1825	TotalRedemptionAmountRule
1826	TotalUnitsNumberRule
1827	TransactionOnAccountOrSubAccountDetailsRule
1828	TransactionOnSubAccountRule
1829	TransactionTypeLegExecutionIdentificationRule
1830	TransactionTypeLegIdentificationRule
1831	TypeAmountRateRule
1832	UserDefinedRule
1833	BilateralBalanceRule
1834	CreditDebitIndicator1Rule
1835	EntryAmountCreditDebitIndicator1Rule
1836	EntryAmountCreditDebitIndicator2Rule
1837	InstructedAmountCreditDebitIndicator1Rule
1838	InstructedAmountCreditDebitIndicator2Rule
1839	InstructedAmountCurrencyRule
1840	PaymentInstructionStatusRule
1841	ReturnCriteriaAndOrSearchCriteriaRule
1842	SettlementAmountCreditDebitIndicator1Rule
1843	SettlementAmountCreditDebitIndicator2Rule
1844	SettlementAmountCurrencyRule
1845	TransactionCreditDebitIndicatorRule
1846	TransferValueDateRule
1847	OrderOriginatorEligibilityGuideline
1848	NomineeAccountServicerRule
1849	TotalSubscriptionAmountRule
1850	AccountIdentificationRule

SWIFTNet Routing Rule

The SWIFTNet Routing Rule is created by you as the Responder, because you are expecting a server request message from SWIFTNet. This rule is used by the SWIFTNet Server adapter to manage interactive messages from the Application SWIFTNet MEFG Server. The rule routes an incoming request message to a user-defined business process based on the following parameters:

- ✦ requestorDN
- ✦ responderDN
- ✦ requestType
- ✦ serviceName

You configure these four parameters and create the SWIFTNet routing rule through the SWIFTNet Routing Rule interface. The SWIFTNet Routing Rule page enables you to assign any business process to a set of Requestor, Responder, Service, and Request Type. The rules are applied to the routing of SWIFTNet messages (Funds, MX, or generic XML format), and the server response messages are constructed and sent back to SWIFTNet through the SWIFTNet MEFG Server.

Note: If you are using CHIPS adapter with SWIFTNet as transport interface, the SWIFTNet Routing Rule will be automatically created for you.

You can also export and import SWIFTNet routing rules.

Creating a SWIFTNet Routing Rule and Associating it with a Business Process

To create a SWIFTNet routing rule and associate it with an appropriate business process to process incoming SWIFTNet requests:

1. If you have not done so already, modify the SWIFTNetClient business process or create a new instance of it. See *SWIFTNetClient Business Process* on page 66.

2. Create a business process to which the inbound messages will be routed by the SWIFTNet Server adapter.

3. From the Application **Deployment** menu, select **Adapter Utilities > SWIFTNet Routing Rule**.

4. To the right of **Create new SWIFTNet routing rule,** click **Go!**.

5. Complete the following parameters and click **Next**:

Parameter	Description
SWIFTNet Routing Rule Name	Type the name of the SWIFTNet routing rule. Required.
Requestor DN	Type the distinguished name of the requestor. Required. **Note:** This value should be taken from the Requestor's SWIFTNet Client service configuration (or from the BPML parameters passed to the service), in accordance with the arrangement made between requestor and responder.

Parameter	Description
Responder DN	Type the distinguished name of the responder. Required. **Note:** This value should be taken from the Requestor's SWIFTNet Client service configuration (or from the BPML parameters passed to the service), in accordance with the arrangement made between requestor and responder.
Service Name	Type the name of the SWIFTNet Client service instance configuration that you created in step 1. Required. **Note:** This value should be taken from the Requestor's SWIFTNet Client service configuration (or from the BPML parameters passed to the service), in accordance with the arrangement made between requestor and responder.
Request Type	Type the request type (message type and name) supported by the file transfer responder. Optional. **Note:** This value should be taken from the Requestor's SWIFTNet Client service configuration (or from the BPML parameters passed to the service), in accordance with the arrangement made between requestor and responder.
Business Process	Select the business process that will be invoked to manage the request and handle the response to a SWIFTNet message (this is the business process that you created in step 2 that will be called when the requestor DN, responder DN, service name, and requestor type match). Required.

6. Click **Finish** to save the routing rule. The rule is now in effect for all incoming SWIFTNet messages (Funds, MX, or generic XML format).

Searching for a SWIFTNet Routing Rule

To edit or delete a SWIFTNet routing rule you must first specify the appropriate rule. You can locate a specific routing rule in two ways:

✦ Search for the routing rule by name.

✦ Select the routing rule from an alphabetical list.

Searching for the routing rule by name is more precise and provides fewer results. Searching from an alphabetical list will result in a list of all routing rules or all rules beginning with a specified letter or digit.

Once you search for the routing rule, you can easily edit or delete it from the SWIFTNet Routing Rule interface.

Searching for a Routing Rule by Name

To search for a routing rule by name:

1. From the Application **Deployment** menu, select **Adapter Utilities > SWIFTNet Routing Rule**.

2. In the Search section, type the name of the routing rule. Case does not matter and you can type part of a name.

   Application returns a list of matches unless no routing rules meet the criteria you specified.

Searching for a Routing Rule from a List

To select a routing rule from a list:

1. From the Application **Deployment** menu, select **Adapter Utilities > SWIFTNet Routing Rule**.

2. In the List section, select one of the following:

    ◆ Alphabetically – Select **All** and click **Go!**

    ◆ Alphabetically – Select a specific letter or digit (0 - 9) and click **Go!**

    Application returns a list of matches unless no routing rules meet your criteria.

Exporting and Importing a SWIFTNet Routing Rule

The Application Import/Export feature enables you to save time and increase the accuracy of duplicating supported resources on different Application environments that are set up for unique purposes. To import and export resources from one Application environment to another Application environment, both environments must be the same version.

Configuring the WebSphere MQ Adapter/Suite to Communicate with SWIFT

The WebSphere MQ adapter and the WebSphere MQ Suite enable you to configure Application to send and receive SWIFT messages to/from SWIFTNet through the WebSphere MQ Interface for SWIFTAlliance Access (MQSA). Also enables you to send and receive messages to/from SWIFTNet though SWIFTNet Alliance Access (SAA) and the SWIFTNet Remote API Host Adapter (RAHA).

This table describes the tasks necessary to configure Application to communicate with SWIFTNet through either the WebSphere MQ adapter or the WebSphere MQ Suite:

Number	Task	For More Information
1	Configure Application to retrieve messages through SAA and MQSA.	*Configuring Application to Retrieve Messages* on page 223
2	Configure Application to send messages through SAA and MQSA.	*Configuring Application to Send Messages* on page 224
3	Disable the UMID and Block S options for all queues within the MQ interface.	*Configuring the UMID and Block S Options* on page 225

Configuring Application to Retrieve Messages

Complete these steps to configure Application to retrieve messages from SWIFTNet through SAA and MQSA:

Note: See *WebSphere MQ Adapter* and *WebSphere MQ Suite* documentation for more details on the general MQ options.

1. Configure an MQ Adapter service instance:

2. Go to **Deployment > Services > Configuration**.

3. Next to New Service, click **Go!**.

4. Select **WebSphere MQ Adapter** and click **Next**.

5. Type a name for the service and a description, and click **Next**.

6. On the "WebSphere MQ Parameters page, type the following information and click **Next**:

   ◆ Set **Host Name** to the name or IP address of the machine hosting the WebSphere MQ that receives messages from SWIFT.

   ◆ Set **Listening Port** to the port number for the MQ installation (if it is something other than the default).

   ◆ Set **Queue Manager** to the name of the queue manager that contains the SWIFTNet message queues.

   ◆ Set **Queue Name** to the name of the queue set up to receive messages from SWIFT.

- ◆ Set **Server Connection Channel** to the name of the connection channel associated with the queue manager.

- ◆ Set **User ID** and **Password** to the login information for the queue that is configured to receive messages, if required.

- ◆ Select the **Receiving messages from WebSphere MQ (Sync)** or **Receiving messages from WebSphere MQ (Async)** option, depending which mode you wish to use.

- ◆ Set the other parameters based on the specific MQ configuration desired.

7. Click **Finish** to save the configuration.

8. Create a business process to retrieve the messages. The following example business process retrieves a message from MQ in synchronous mode, assuming the service configuration **FromSAA** was created in step 1. Optional parameters have been set to make the business process wait for up to 10 seconds for a message to appear before timing out:

```
<process name="FromSAABP">
    <operation name="WebSphere MQ Adapter">
        <participant name="FromSAA"/>
        <output message="WebsphereMQInputMessage">
            <assign to="." from="*"></assign>
            <assign to="rcv_MQGMO_wait">Yes</assign>
            <assign to="rcv_MQGMO_waitInterval">10000</assign>
        </output>
        <input message="inmsg">
            <assign to="." from="*"></assign>
        </input>
    </operation>
</process>
```

Configuring Application to Send Messages

Complete the following steps to configure Application to send messages to SWIFTNet through SAA and MQSA:

1. Configure an MQ Adapter service instance:

2. Go to **Deployment > Services > Configuration**.

3. Next to New Service, click **Go!**.

4. Select **WebSphere MQ Adapter** and click **Next**.

5. Type a name for the service and a description, and click **Next**.

6. On the "WebSphere MQ Parameters page, type the following information and click **Next**:

- ◆ Set **Host Name** to the name or IP address of the machine hosting the WebSphere MQ that receives messages from SWIFT.

- ◆ Set **Listening Port** to the port number for the MQ installation (if it is something other than the default).

- ◆ Set **Queue Manager** to the name of the queue manager that contains the SWIFTNet message queues.

- ◆ Set **Queue Name** to the name of the queue set up to receive messages from SWIFT.

- ◆ Set **Server Connection Channel** to the name of the connection channel associated with the queue manager.

- ◆ Set **User ID** and **Password** to the login information for the queue that is configured to receive messages, if required.

- ◆ Select the **Sending messages to WebSphere MQ** option.

- ◆ Set the other parameters based on the specific MQ configuration desired.

7. Click **Finish** to save the configuration.

8. Create a business process to send the message. The following example business process sends a message from MQ in synchronous mode, assuming the service configuration **ToSAA** was created in step 1. .

```
<process name="ToSAABP">
    <operation name="WebSphere MQ Adapter">
      <participant name="ToSAA"/>
      <output message="WebsphereMQInputMessage">
        <assign to="." from="*"></assign>
        <assign to="snd_MQMD_msgType">DATAGRAM</assign>
      </output>
      <input message="inmsg">
        <assign to="." from="*"></assign>
      </input>
    </operation>
</process>
```

Configuring the UMID and Block S Options

Application does not support the UMID and S block options, so those must be disabled for all Queues within the MQSeries Interface.

Complete these steps to disable the **UMID** and **Block S** options:

1. Open the SWIFTAlliance Workstation application.

2. Double-click the **MQSeries Interface**.

3. Open each queue and verify the following options:

- ◆ For both From MQ and To MQ queues, **Include UMID** must be set to No.

- ◆ For the To MQ queues, **Include Block S** must be set to Without Block S.