SWIFTNet Server Adapter

The SWIFTNet Server adapter communicates to the SWIFTNet Network through the SWIFTNet MEFG server. It responds to and accepts InterAct and FileAct messages that are sent by remote SWIFTNet correspondents. The following table provides an overview of the SWIFTNet Server adapter:

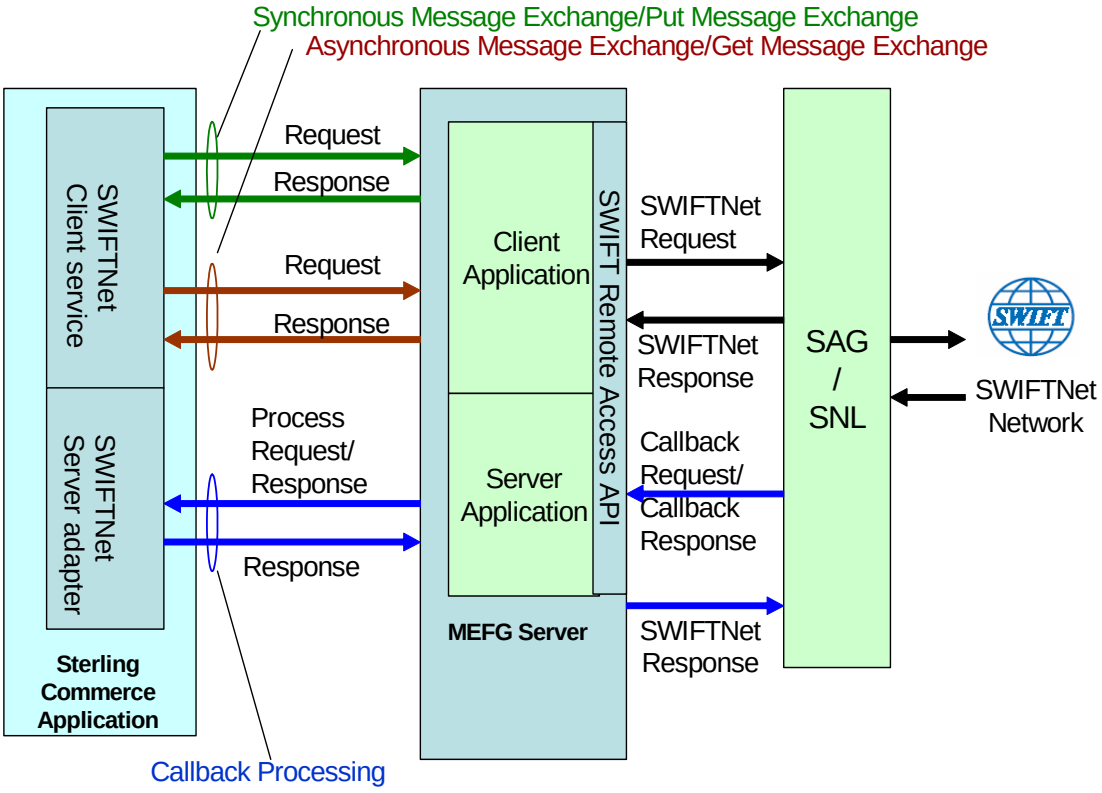
System Name	SWIFTNet Server Adapter
Graphical Process Modeler (GPM) categories)	All services
Description	This adapter is responsible for receiving and responding to SWIFTNet InterAct and FileAct messages using the SWIFTNet MEFG Server.
Business usage	A business would use this adapter in order to exchange SWIFTNet InterAct and FileAct messages with its trading partners over the SWIFTNet system. **Note:** You will also need to configure this adapter if you are transporting CHIPS messages using the SWIFTNet transport mode. This adapter also sends acknowledgements to CHIPS.
Usage example	When an InterAct or FileAct message is received, a business process is executed to process the message and, when required, to generate the SWIFTNet response.
Preconfigured?	This adapter is preconfigured as part of the application installation.
Requires third party files?	No third party files are required.
Platform availability	All supported application platforms.
Related services	This is designed to work in conjunction with the SWIFTNet MEFG Server and the Command Line Adapter 2. This service also works with the SWIFTNet HTTP Server adapter to provide SSL support.
Application requirements	SSL can be implemented between the application and the MEFG Server if the SWIFTNet HTTP Server adapter is configured for that setup.
Initiates business processes?	Initiates system business processes.
Invocation	By the Multi-Enterprise Financial Gateway for SWIFTNet application.
Business process context considerations	None.
Returned status values	◆ Fatal—non-recoverable error ◆ Transient—recoverable error ◆ Logic—recoverable error ◆ Success—Success ◆ Warning—Success with warning
Restrictions	Only one SWIFTNet MEFG Server can be configured to talk to one SWIFTNet Server adapter instance in the application.

Persistence level	N/A
Testing considerations	N/A

How the SWIFTNet Server Adapter Works

The SWIFTNet Server adapter is comprised of two parts: the service part and the adapter part. The service part is used in a business process 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 the application using the Command Line Adapter 2 (CLA2), which is built into the SWIFTNet Server adapter. Starting and stopping the operation of the SWIFTNet MEFG Server will only work correctly if the CLA2Client.jar is deployed in the same machine where the SWIFTNet MEFG Server is installed. The CLA2Client.jar file must also be started by a user who has permission to access the SWIFTNet MEFG Server home directory.

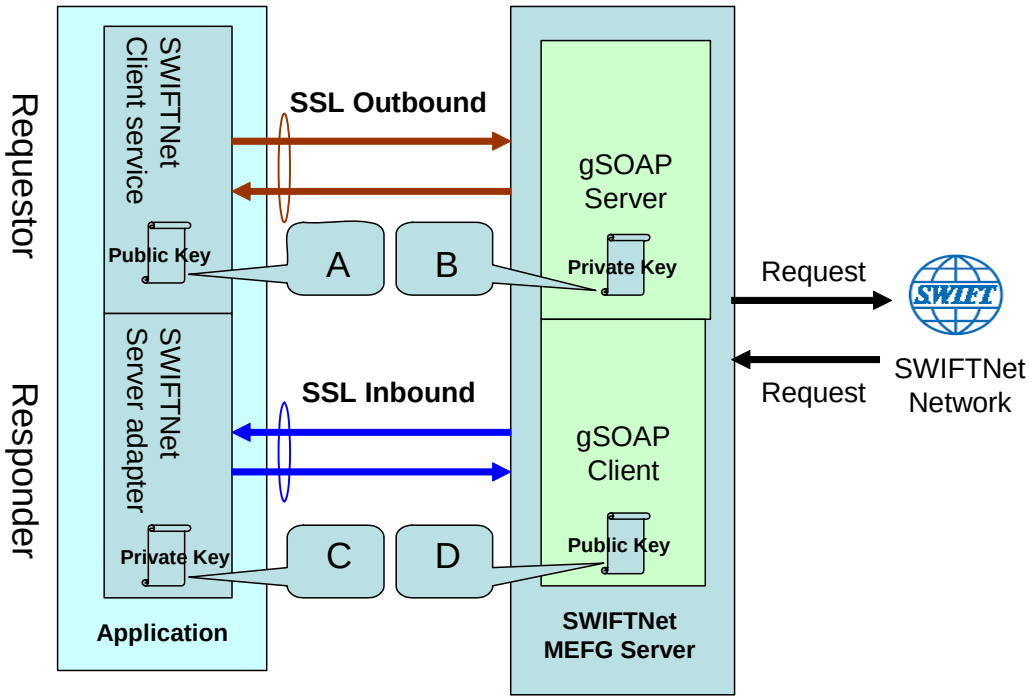
This diagram illustrates the process flow between the application and the SWIFTNet network through the SWIFTNet MEFG Server (not using SSL):



The SWIFTNet Server adapter (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 the

application, two channels are secured: an Outbound channel (the application acting as the Requestor) and an Inbound channel (the application acting as the Responder).

This diagram illustrates the configuration necessary between the application and the SWIFTNet network through the SWIFTNet MEFG Server to set up 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 the 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 subdirectory). The sslUtil.jar file is located in the bin subdirectory of the SWIFTNet MEFG Server installation 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 the 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 directory).

Note: To configure SSL on the SWIFTNet MEFG Server, run the following command in the bin directory of the MEFG SWIFTNet Server installation bin sub-directory.:

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

Note: The application enables you easily renew certificates. See *Renewing a Certificate* on page 16 for more information.

Implementing the SWIFTNet Server Adapter

To implement the SWIFTNet Server adapter, complete the following tasks:

1. Create a configuration of the Command Line Adapter 2.
   a. Locate the client jar (CLA2Client.jar) that contains the necessary classes.
   b. Move the client jar to the machine where you will be running the remote adapter.
   c. Start the remote adapter using the following command:

      ```
      java -jar CLA2Client.jar <port> [debug]
      ```

Note: The [debug] option is not required, but is provided for your convenience. If you upgrade the application, you may need to obtain a new CLA2Client.jar file to avoid a Class Conflict error.

2. Create a configuration of the SWIFTNet Server adapter. See *Managing Adapters and Services*. For information about the fields specific to this adapter, see *Configuring the SWIFTNet Server Adapter* on page 5.
3. Specify field settings for the adapter configuration in the application Admin Console and in the GPM as necessary. See *Creating or Setting Up a Adapter Configuration in the Admin Console* on page 5 or *Setting Up the Adapter in the GPM* on page 13.
4. Configure the business process you are using for the SWIFTNet Server adapter.

   The business processes that work with SWIFTNet Server adapter include the following:

   - handleSWIFTNetServerSnFRequest
   - handleSWIFTNetInboundCorrelation
   - handleSWIFTNetOutboundCorrelation
   - handleSWIFTNetServerFADelNotif
   - handleSWIFTNetServerFAEvent
   - handleSWIFTNetServerFARequest
   - handleSWIFTNetServerFASnFDelNotif
   - handleSWIFTNetServerFASnFRequest
   - handleSWIFTNetServerRequest
   - handleSWIFTNetServerSnFDelNotif
   - handleSWIFTNetServerFASnFEvent
   - handleSWIFTNetSnFInboundCorrelation
   - handleSWIFTNetSnFOutboundCorrelation
5. Define the **HTTP Listen Port** in the SWIFTNet HTTP Server adapter instance, which should have the same value as the **GIS HTTP Sever Adapter Port** defined in the SWIFTNet Server adapter configuration.

6. Specify field settings in the business process. See *Business Process Example* on page 16.

Configuring the SWIFTNet Server Adapter

1. Select **Deployment > Services > Configuration**.
2. Search for SWIFTNet Server adapter or select it from the list and click **Go!**.
3. Click **Edit**.
4. Specify field settings in the Admin Console (*Creating or Setting Up a Adapter Configuration in the Admin Console* on page 5)—alternatively you can specify field settings in the GPM (*Setting Up the Adapter in the GPM* on page 13), but you will need to access the adapter instance through the Admin console to enable the instance (as described in step 5).

Note: Specify failover processing to ensure that failover is supported if a SAG connection fails by configuring **Active-Active Configuration**.

Note: For specific instructions on configuring an input channel, see *SWIFT Input Channel*.

5. After configuring the SWIFTNet Server adapter in the Admin Console, click the **Enable Service for Business Process** check box on the Confirm page to enable the instance.
6. Once the SWIFTNet Server adapter is configured and saved, click the **Enabled** check box on the Services Configuration page. This starts the SWIFTNet MEFG Server.
7. On the Confirm page, verify that the **Enable Service for Business Processes** check box is selected to enable the adapter instance.

You must specify field settings in the application, using the Admin Console and the GPM.

Creating or Setting Up a Adapter Configuration in the Admin Console

Use the field definitions in the following table to create a new configuration of the SWIFTNet Server adapter, or to set up the configuration provided with the application. Some fields are available in both the Admin Console and in the GPM.

Note: The business entities (accessible through the Business Entities wizard as part of the SWIFTNet Server adapter configuration) are shared by both RA1 and RA2. The Business Entities wizard enables you to add multiple entities.

Field	Description
Name	Unique and meaningful name for the service configuration. Required.
Description	Meaningful description for the service configuration, for reference purposes. Required.

Field	Description
Select a Group	Select one of the options: ◆ None – Do not include the configuration in a service group at this time. ◆ Create New Group – Enter a unique name for a new group, which will be created with this configuration. (You can then add other services to the group as well.) ◆ Select Group – If service groups already exist for this service type, they are displayed in the list. Select a group from the list. **Note:** Only select group if this adapter is clustered in a group. See *Managing Services and Services*.
GIS Server IP	The callback IP of the application for the SWIFTNet MEFG Server. Required. **Note:** The default value is the IP address of the machine where the application is installed.
GIS HTTP Server Adapter Port	This is the listening port for the SWIFTNet HTTP Server Adapter. Required. The default populated value is the instance port number of the application instance plus 53. For example, if the application instance port is 34600, the listening port populated by default is 34653. **Note:** The HTTP Server adapter functions between the SWIFTNet Server adapter and the SWIFTNet MEFG Server. For an SSL connection, this value should be server name because the certificate is made with the server name. **Note:** If you are using the SWIFTNet Server adapter in your current installation, prior to installing a new version of the Standards Library, you need to note the value you have configured for this parameter. This parameter may be overwritten during the upgrade process (replaced with the default value). If this parameter is overwritten, you need to restore it to the original value after the upgrade process is complete.
MEFG SWIFTNet IP	The IP address of the SWIFTNet MEFG Server. Required.
MEFG SWIFTNet Port	The port of the SWIFTNet MEFG Server. Required.
CLA2Client Listening Port	The listening port used by the client command adapter (CLA2Client) running along the SWIFTNet MEFG Server. Required. **Note:** This port listens for requests to start and stop the SWIFTNet MEFG Server.
MEFG SWIFTNet Home	The home directory of the SWIFTNet MEFG Server. Required.
Use SSL	Whether to enable Secure Sockets Layer (SSL) over HTTP communication between the application and the SWIFTNet MEFG Server. Valid values are False (default) and True.
Cipher Strength	Specifies the strength of the algorithms (cipher suites) used to encrypt data. Valid values are: ◆ STRONG - Required if Use SSL is Must ◆ ALL - All cipher strengths are supported ◆ WEAK - Often required for international trade, because government regulations prohibit STRONG encryption from being exported Default is ALL. Required if SSL is checked.
CA Certificate	Move one or more CA Certificates to the use column. These are the digital security certificates that the SSL server will use to authenticate the client. Optional.

Field	Description
Message Partner Client Name	The client message partner name that the SNL server application recognizes for the SWIFTNet MEFG Server client application. **Note:** The Message Partner Client Name must correspond to the Application Interface Message Partner that is defined on the SAG as the client interface for the SWIFTNet MEFG Server.
Message Partner Server Name	The server message partner name that the SNL server application recognizes for the SWIFTNet MEFG Server server application. **Note:** The Message Partner Server Name must correspond to the Application Interface Message Partner that is defined on the SAG as the client interface for the SWIFTNet MEFG Server.
Active-Active Configuration	Enables you to set up active-active configuration using two separate instances of the Remote API (RA), RA1 and RA2. Each RA should be configured to point to a different SAG to support failover processing. Possible values are True and False (default). Required. **Note:** This parameter specifies whether to support failover if one SAG fails. When this parameter is set to True, you are presented with parameters for both an RA1 Profile and an RA2 Profile. When you are operating in an environment with multiple SAGs configured in active-active mode, setting this parameter enables you to define an alternate RA connection to a secondary SAG for failover support.
SNL Endpoint (for Store and Forward only)	The SNL endpoint used to receive data from SnF queues (for example, snl_sft). Optional—complete only if using store and forward processing. **Note:** You must define endpoints on the SAG to route the InterAct messages to the correct application interface. If you are using store-and-forward, an extra endpoint is required to route messages coming from the store-and-forward queue (you can use the default endpoint for store-and-forward, **snl_sft**).
SnF Monitoring Interval (in seconds)	The store and forward monitoring interval (in seconds). Optional. **Note:** This parameter enables you to indicate the interval that you want the SWIFTNet MEFG Server to check on the queue status. The SWIFTNet MEFG Server sets a timer to send the GetSnFStatusRequest message based on the value you enter.
Return Signature List	Whether you want your own signature returned. Valid values are False (default - normal Crypto is used) and True. Optional for T-Copy and Y-Copy implementation.
Use Input Channel (for InterAct Store and Forward only)	Whether to use the input channel with this adapter. Valid values are False (default) and True. You have to select True if you are using an input channel. Required. **Note:** Used for InterAct store-and-forward only. If you configure this parameter, the SWIFTNet MEFG Server opens the Input Channel automatically during the startup (when the SWIFTNet Server Adapter is enabled). This Input Channel remains open until the SWIFTNet MEFG Server is shut down (or the SWIFTNet Server Adapter is disabled). During this time, you still have an option to send message using the input channel or without the input channel. All you need to do is to indicate this by using this parameter in SWIFTNet Client service.

Field	Description
SWIFTNet RA	The absolute path of the RA1 installation directory for RA1 SWIFTNet. Required. For example, **/SWIFTAlliance/RA**. **Note:** This parameter specifies where to pick up the remote API and execute to SAG.
Config	The relative path of the RA1 instance configuration directory (relative to the RA installation directory). Required. For example, **RA1/cfg**. **Note:** If you are using the SWIFTNet Server adapter in your current installation, prior to installing a new version of the Standards Library, you need to note the value you have configured for this parameter. This parameter may be overwritten during the upgrade process (replaced with the default value). If this parameter is overwritten, you need to restore it to the original value after the upgrade process is complete.
Bin	This is added to the PATH environment variable to contain the SWIFTNet MEFG Server binaries. Possible value is bin. Required. **Note:** If you are using the SWIFTNet Server adapter in your current installation, prior to installing a new version of the Standards Library, you need to note the value you have configured for this parameter. This parameter may be overwritten during the upgrade process (replaced with the default value). If this parameter is overwritten, you need to restore it to the original value after the upgrade process is complete.
Lib	This is added to the library path environment variable. Possible value is lib. Required. **Note:** If you are using the SWIFTNet Server adapter in your current installation, prior to installing a new version of the Standards Library, you need to note the value you have configured for this parameter. This parameter may be overwritten during the upgrade process (replaced with the default value). If this parameter is overwritten, you need to restore it to the original value after the upgrade process is complete.
Category	This is the category of RA. Possible values are: ◆ RA (SNL facade library to access an SAG) ◆ SNL (a native SNL interface) ◆ DEFAULT (default set for the RA1 instance) Required. **Note:** If you are using the SWIFTNet Server adapter in your current installation, prior to installing a new version of the Standards Library, you need to note the value you have configured for this parameter. This parameter may be overwritten during the upgrade process (replaced with the default value). If this parameter is overwritten, you need to restore it to the original value after the upgrade process is complete.
Delivery Notification	Determines whether the RA1 server is handling a delivery notification. Possible values are True and False (default). Optional. This is used for a FileAct get.
Delivery Notification DN	Distinguished name of the responder of the delivery notification. Optional.
Request Type of Del. Notifn	Request type of the delivery notification. This is used for a FileAct Get. Required.
Send Del. Notifn before Backend Processing	Indicates if the server will send a delivery notification before the internal process is executed. Required.

Field	Description
Event Status Tracking	Indicates if the server requires all the FileAct Event statuses to be returned. Valid values are: ◆ Minimal (only Completed, Rejected, Duplicated statuses will be returned) ◆ Full (all statuses are returned) Required.
SWIFTNet RA	The absolute path of the RA2 installation directory for RA2 SWIFTNet. Required (based on Active-Active configuration). For example, **/SWIFTAlliance/RA**. **Note:** This parameter is only displayed if **Active-Active Configuration** is set to True.
Config	The relative path of the RA2 instance configuration directory (relative to the RA2 installation directory). Required (based on Active-Active configuration). For example, **/RA2/cfg**. **Note:** This parameter is only displayed if **Active-Active Configuration** is set to True. **Note:** If you are using the SWIFTNet Server adapter in your current installation, prior to installing a new version of the Standards Library, you need to note the value you have configured for this parameter. This parameter may be overwritten during the upgrade process (replaced with the default value). If this parameter is overwritten, you need to restore it to the original value after the upgrade process is complete.
Bin	This is added to the PATH environment variable to contain the SWIFTNet MEFG Server binaries. Required (based on Active-Active configuration). **Note:** This parameter is only displayed if **Active-Active Configuration** is set to True. **Note:** If you are using the SWIFTNet Server adapter in your current installation, prior to installing a new version of the Standards Library, you need to note the value you have configured for this parameter. This parameter may be overwritten during the upgrade process (replaced with the default value). If this parameter is overwritten, you need to restore it to the original value after the upgrade process is complete.
Lib	This is added to the library path environment variable. Required (based on Active-Active configuration). **Note:** This parameter is only displayed if **Active-Active Configuration** is set to True. **Note:** If you are using the SWIFTNet Server adapter in your current installation, prior to installing a new version of the Standards Library, you need to note the value you have configured for this parameter. This parameter may be overwritten during the upgrade process (replaced with the default value). If this parameter is overwritten, you need to restore it to the original value after the upgrade process is complete.

Field	Description
Category	This is the category of RA2. Possible values are: ◆ RA (SNL facade library to access an SAG) ◆ SNL (a native SNL interface) ◆ DEFAULT (default set for the RA1 instance) Required (based on Active-Active configuration). **Note:** This parameter is only displayed if **Active-Active Configuration** is set to True. **Note:** If you are using the SWIFTNet Server adapter in your current installation, prior to installing a new version of the Standards Library, you need to note the value you have configured for this parameter. This parameter may be overwritten during the upgrade process (replaced with the default value). If this parameter is overwritten, you need to restore it to the original value after the upgrade process is complete.
Delivery Notification	Determines whether the RA2 server is handling a delivery notification. Possible values are True and False (default). Optional. This is used for a FileAct get.
Delivery Notification DN	Distinguished name of the responder of the delivery notification. Optional.
Request Type of Del. Notifn	Request type of the delivery notification. This is used for a FileAct Get. Required.
Send Del. Notifn before Backend Processing	Indicates if the server will send a delivery notification before the internal process is executed. Required.
Event Status Tracking	Indicates if the server requires all the FileAct Event statuses to be returned. Valid values are: ◆ Minimal (only Completed, Rejected, Duplicated statuses will be returned) ◆ Full (all statuses are returned) Required.
Input Channel Name	The name of the input channel. Required only if you specified **True** for **Use Input Channel**.
Authoriser DN	The authorized distinguished name that will be used to open the input channel. Required only if you specified **True** for **Use Input Channel**.
Force Open the Input Channel	Whether to force open the input channel or use normal mode. Valid values are False (use normal mode, which is the default) and True (force the input channel). Required only if you specified **True** for **Use Input Channel**.
Max. Resend Attempts	The maximum number of resend attempts allowed before the application automatically sends a Resolve Gap request to SWIFT. The default is 3. Required only if you specified **True** for **Use Input Channel**.
Run As User	Identify a user who has permission to run the scheduled activity. You can type the user ID, click the button to select the user ID from the list, and click **Save**. Optional. **Note:** You must configure the parameters on the Schedule Type page for the Resend Scheduler to work correctly.

Field	Description
Use 24 Hour Clock Display	By default, the scheduling wizard displays times using a 12-hour clock (which designates the time in hours as a.m. or p.m.). Use this option to display times using a 24-hour clock. Optional. **Note:** We recommend that you set the Resend Scheduler to **Run based on timer** and set it for one minute under normal usage (that is, every 1 Mins(s)). You must configure the parameters on the Schedule Type page for the Resend Scheduler to work correctly.
Do not use schedule	Removes all references to a schedule from the service. If you select this option, you cannot enable the schedule in the future. You must recreate the schedule instead. Use this option only when you do not need a schedule for the service. This is the default option. Optional. **Note:** We recommend that you set the Resend Scheduler to **Run based on timer** and set it for one minute under normal usage (that is, every 1 Mins(s)). You must configure the parameters on the Schedule Type page for the Resend Scheduler to work correctly.
Run based on timer	Run the service at a certain time or time interval, such as every 2 hours. Optional. **Note:** We recommend that you set the Resend Scheduler to **Run based on timer** and set it for one minute under normal usage (that is, every 1 Mins(s)). You must configure the parameters on the Schedule Type page for the Resend Scheduler to work correctly.
Select Time	Type the time at which you want the Resend Scheduler to run. **Note:** We recommend that you set the Resend Scheduler to **Run based on timer** and set it for one minute under normal usage (that is, every 1 Mins(s)). You must configure the parameters on the Schedule Type page for the Resend Scheduler to work correctly.
Run daily	Run the service one or more times every day. Optional. **Note:** We recommend that you set the Resend Scheduler to **Run based on timer** and set it for one minute under normal usage (that is, every 1 Mins(s)). You must configure the parameters on the Schedule Type page for the Resend Scheduler to work correctly.
Run based on days of the week	Run the service on certain days of the week, such as every Monday. Optional. **Note:** We recommend that you set the Resend Scheduler to **Run based on timer** and set it for one minute under normal usage (that is, every 1 Mins(s)). You must configure the parameters on the Schedule Type page for the Resend Scheduler to work correctly.
Run based on days of the month	Run the service on certain days of the month, such as the 1st or 15th of every month. Optional. **Note:** We recommend that you set the Resend Scheduler to **Run based on timer** and set it for one minute under normal usage (that is, every 1 Mins(s)). You must configure the parameters on the Schedule Type page for the Resend Scheduler to work correctly.
Schedule Exclusions	Allows you to add any schedule anomalies (when the Resend Scheduler should not run). **Note:** We recommend you leave this parameter blank (that is, do not create any schedule exclusions).

Field	Description
Date Exclusions	Allows you to add any date anomalies (any date on which the Resend Scheduler should not run). **Note:** We recommend you leave this parameter blank (that is, do not create any date exclusions).
New Business Entity	Click **add** to create a new business entity or click **edit** to modify an existing entity. **Note:** You must have at least one business entity created to proceed.
Entity	Identifies the security context to be used. For the client, the business entity is the requester. For the server, the business entity is the responder. Required for each configured entity to access a proprietary SWIFTNet PK1 certificate to set up a valid security context. **Note:** This is the distinguished name created by SWIFT. This parameter is only displayed if you edit an existing Business Entity or add a new Business Entity. The business entities are shared by both the RA1 and RA2 profiles.
UserId	The user identifier for this business entity (to log in to SWIFTNet). Required for each configured entity. **Note:** The UserName is created in SAG (in the Users Module) and must also have a certificate created for it in the SAG. This parameter is only displayed if you edit an existing Business Entity or add a new Business Entity.
Password	The user password for this business entity (to log in to SWIFTNet). Required for each configured entity. **Note:** This password is automatically encrypted. This parameter is only displayed if you edit an existing Business Entity or add a new Business Entity.
Message Queue	The name of the store and forward queue from which to receive messages. Required in Store and Forward mode.
Notification Queue	The Name of the store-and-forward queue to retrieve delivery notifications (optional; if empty, same as Message Queue). Required in Store and Forward mode.
Acquire queue by force	Whether to acquire the queue by force. Valid values are False (default) and True. Required.
Use Default Delivery Notification	Indicates whether to use the default delivery notification configuration on the RA1 page. Required.
Delivery Notification (Del. Notifn)	Indicates whether the sender asked the receiver to send a delivery notification. Optional. Valid values are True (default) or False. **Note:** This parameter is only available when **Use Default Delivery Notification** is not selected.
Request Type of Del. Notifn	If **Delivery Notification (Del. Notifn)** is set to True, the value of this parameter is used to request a specific delivery notification message from the remote receiving server application when it returns the delivery notification. Optional. **Note:** This parameter is only available when **Use Default Delivery Notification** is not selected.
Reception Directory	The full directory path where the file is received and stored during FileAct Put mode. Required for FileAct. Optional.
Download Directory	The full directory path where the file is picked up and sent to the requestor during FileAct Get mode. Required for FileAct. Optional.

Field	Description
Success Directory	The full directory path that must be specified when using the FileAct #OLDEST_FILE feature. Required for FileAct. Optional.

Setting Up the Adapter in the GPM

Use the field definitions in the following table to set up the adapter configuration in the GPM:

Field	Description
Active-Active Configuration	Enables you to set up active-active configuration using two separate instances of the Remote API (RA), RA1 and RA2. Each RA should be configured to point to a different SAG to support failover processing. Possible values are True and False (default). Required. **Note:** This parameter specifies whether to support failover if one SAG fails. When this parameter is set to True, you are presented with parameters for both an RA1 Profile and an RA2 Profile. When you are operating in an environment with multiple SAGs configured in active-active mode, setting this parameter enables you to define an alternate RA connection to a secondary SAG for failover support.
commandLinePort	The listening port used by the client command adapter (CLA2Client) running along the SWIFTNet MEFG Server. Required. **Note:** This port listens for requests to stop the SWIFTNet MEFG Server.
deliveryNotification	Determines whether the server is handling a delivery notification. Possible values are True and False (default). Optional. **Note:** This is a BPML parameter used by the SWIFTNet Server adapter to construct the response back to the SWIFTNet MEFG Server.
Description	Error description for the rejected response. Optional. **Note:** This is a BPML parameter used by the SWIFTNet Server adapter to construct the response back to the SWIFTNet MEFG Server.
downloadDir	The full directory path where the file is picked up and sent to the requestor during FileAct Get mode. Required for FileAct.
Info	Error information for the rejected response. Optional. **Note:** This is a BPML parameter used by the SWIFTNet Server adapter to construct the response back to the SWIFTNet MEFG Server.
interfaceMode	SWIFTNet message type. Valid values are InterAct or FileAct. **Note:** This is a BPML parameter used by the SWIFTNet Server adapter to construct the response back to the SWIFTNet MEFG Server.
localServerAddress	The callback IP of the application for the SWIFTNet MEFG Server. Required.
localServerPort	The is the listening port for the SWIFTNet HTTP Server adapter. Required. **Note:** The HTTP Server adapter functions in between the SWIFTNet Server adapter and the SWIFTNet MEFG Server.

Field	Description
messageID	Message identifier for the incoming message. Required. **Note:** This is a unique identifier. This is a BPML parameter used by the SWIFTNet Server adapter to construct the response back to the SWIFTNet MEFG Server.
messagePartnerClient Name	The client message partner name that the SNL server application recognizes for the SWIFTNet MEFG Server client application.
messagePartnerServer Name	The server message partner name that the SNL server application recognizes for the SWIFTNet MEFG Server server application. **Note:** The Message Partner Server Name must correspond to the Application Interface Message Partner that is defined on the SAG as the client interface for the SWIFTNet MEFG Server.
RA1Bin	This is added to the PATH environment variable. Possible value is bin. Required.
RA1Category	This is the category of RA. Possible values are: ◆ RA (SNL facade library to access an SAG) ◆ SNL (a native SNL interface) ◆ DEFAULT (default set for the RA1 instance) Required.
RA1Config	The relative path of the RA1 instance configuration directory (relative to the RA installation directory). Required.
RA1deliveryNotification	Determines whether the RA1 server is handling a delivery notification. Possible values are True and False (default). Optional.
RA1deliveryNotification DN	Distinguished name of the responder of the delivery notification. Optional.
RA1deliverynotification RT	Request type of the delivery notification. This is used for a FileAct Get. Required.
RA1eventstatusTracking	Indicates if the server requires all the FileAct Event statuses to be return. Valid values are: ◆ Minimal (only Completed, Rejected, Duplicated statuses will be returned) ◆ Full (all statuses are returned) Required.
RA1Lib	This is added to the library path environment variable. Possible value is lib. Required.
RA1sendDNb4bkend Process	Indicates if the server will send a delivery notification before the internal process is executed. Required.
RA1Swiftnethome	The home directory of the SWIFTNet MEFG Server. Required. **Note:** This is an absolute path location. This parameter specifies where to pick up the remote API and execute to SAG.
RA2Bin	This is added to the PATH environment variable. Possible value is bin. Required. **Note:** This parameter is only displayed if **Active-Active Configuration** is set to True.

Field	Description
RA2Category	This is the category of RA. Possible values are: ◆ RA (SNL facade library to access an SAG) ◆ SNL (a native SNL interface) ◆ DEFAULT (default set for the RA1 instance) Required. **Note:** This parameter is only displayed if **Active-Active Configuration** is set to True.
RA2Config	The relative path of the RA2 instance configuration directory (relative to the RA installation directory). Required.
RA2deliveryNotification	Determines whether the RA2 server is handling a delivery notification. Possible values are True and False (default). Optional.
RA2deliveryNotification DN	Distinguished name of the responder of the delivery notification. Optional.
RA2deliverynotification RT	Request type of the delivery notification. This is used for a FileAct Get. Required.
RA2eventstatusTracking	Indicates if the server requires all the FileAct Event statuses to be return. Valid values are: ◆ Minimal (only Completed, Rejected, Duplicated statuses will be returned) ◆ Full (all statuses are returned) Required.
RA2Lib	This is added to the library path environment variable. Possible value is lib. Required.
RA2sendDNb4bkend Process	Indicates if the server will send a delivery notification before the internal process is executed. Required.
RA2Swiftnethome	The home directory of the RA2. Required. **Note:** This parameter is only displayed if **Active-Active Configuration** is set to True. **Note:** This is an absolute path location. This parameter specifies where to pick up the remote API and execute to SAG.
receptionDir	The full directory path where the file is received and stored during FileAct Put mode. Required for FileAct.
remoteServerAddress	The IP address of the SWIFTNet MEFG Server. Required.
remoteServerPort	The port of the SWIFTNet MEFG Server. Required.
SnF	Indicates if you are using the store-and-forward method. Valid values are True (use Store-and-Forward) and False (default—do not use Store-and-Forward). Required. **Note:** This is a BPML parameter used by the SWIFTNet Server adapter to construct the response back to the SWIFTNet MEFG Server.
snlEndPoint	The SNL endpoint used to receive data from SnF queues (for example, snl_sft). Optional—complete only if using store and forward processing.

Field	Description
Status	The status of the message. Possible values are: ◆ Accepted ◆ Rejected ◆ Failed ◆ Duplicated Required. **Note:** This is a BPML parameter used by the SWIFTNet Server adapter to construct the response back to the SWIFTNet MEFG Server.
successDir	The full directory path that must be specified when using the FileAct #OLDEST_FILE feature. Required for FileAct.
swiftnetServerHome Directory	The home directory where the SWIFTNet MEFG Server is installed. Required.

Business Process Example

The service part of the SWIFTNet Server adapter that is used in the business process is bootstrapped when the SWIFTNet MEFG Server posts the request through the URI defined in the HTTP Server adapter. For more information about the HTTP Server adapter, see *HTTP Server Adapter*.

Renewing a Certificate

You can create a business process and schedule it to be executed at an interval of three months. You only need to pass in the distinguished name that is specified in the SWIFTNet Server Adapter Business Entities. When the request is passed to the SWIFTNet MEFG Server, it looks up the user identifier and the encrypted password in the configuration file. The SWIFTNet MEFG Server then performs an initRequest and CreateSecurityContext to open the certificate.

```
<operation>

   <participant name="SWIFTNetClientService"/>

      <output message="renewSecurityContext">

      <assign to="renewDN">o=yourDN,o=swift</assign>

      <assign to="." from="*"/>

   </output>

   <input message="testing">

      <assign to="." from="*"/>

   </input></operation>
```

Interact Business Process Without Store-and-Forward Processing

The following business process example (in which the service part of the SWIFTNet Server adapter as part of InterAct processing) is used if you are not using store-and-forward processing:

Note: This business process is from 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>
```

InterAct Business Process With Store-and-Forward Processing

The following business process example demonstrates the service part of the SWIFTNet Server adapter being used as part of InterAct processing if you are using store-and-forward processing:

Note: This business process is from the handleSWIFTNetServerSnFRequest business process.

```
<process name="handleSWIFTNetServerSnFRequest">
  <rule name="IsAuthNotification">
    <condition>SwiftServerRequest/AuthResponse = 'TRUE'</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>
    <choice name="AddToMailbox">
      <select>
        <case ref="IsAuthNotification" negative="true" activity="Mailbox Add
Service"/>
      </select>
      <!-- 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 -->
      <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>
    </choice>
    <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>
```

Fileact Business Process Without Store-and-Forward Processing

The following business process example shows the service part of the SWIFTNet Server adapter as part of FileAct processing without using store-and-forward processing:

Note: This business process is from the handleSWIFTNetServerRequest 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>
```

FileAct Business Process With Store-and-Forward Processing

The following business process example shows the service part of the SWIFTNet Server adapter used as part of FileAct processing with store-and-forward processing:

Note: This business process is from the handleSWIFTNetServerFASnFRequest business process.

```
<process name="handleSWIFTNetServerFASnFRequest">
  <rule name="UndefinedCopyOrForceReject">
    <condition>SwiftServerRequest/AuthRequest = 'N' or
SwiftServerRequest/FileInfoForceMode = 'Rejected'</condition>
  </rule>
  <rule name="AuthorizationNeeded">
    <condition>SwiftServerRequest/AuthRequest = 'Y' and
SwiftServerRequest/FileInfoForceMode != 'Refused'</condition>
  </rule>
  <rule name="ForceRefusal">
    <condition>SwiftServerRequest/FileInfoForceMode = 'Refused'</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>
    <choice name="NeedAuthorization">
      <select>
        <case ref="AuthorizationNeeded" activity="Mailbox Add Service"/>
      </select>
     <!-- Put into a Mailbox so that it can bootstrap internal authorization business
process later -->
      <!-- Mailbox path is based on SwiftServerRequest/recipientDN/requestorDN/ -->
      <operation name="Mailbox Add Service">
        <participant name="MailboxAdd"/>
        <output message="AddRequest">
          <assign to="." from="*"/>
          <assign to="PrimaryDocument" from="HeaderInfo/@SCIObjectID"/>
          <assign to="MessageName" from="concat('ThirdParty_',
SwiftServerRequest/copySnFReference/text())"/>
          <assign to="MailboxPath" from="concat('/',
SwiftServerRequest/recipientDN/text(),'/',SwiftServerRequest/requestorDN/text())"/>
          <assign to="ExtractableCount">1</assign>
          <assign to="ContentType">ascii</assign>
        </output>
        <input message="inmsg">
          <assign to="AddResults" from="*"/>
        </input>
      </operation>
    </choice>
    <choice name="IsUndefinedCopyOrForceReject">
      <select>
```

```
        <case ref="UndefinedCopyOrForceReject" negative="true"
activity="AcceptRequest"/>
        <case ref="UndefinedCopyOrForceReject" activity="RejectRequest"/>
      </select>
      <operation name="AcceptRequest">
        <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>
      <sequence name="RejectRequest">
        <operation name="ReleasePrimDoc">
          <participant name="ReleaseService"/>
          <output message="outmsg">
            <assign to="TARGET">/ProcessData/PrimaryDocument</assign>
            <assign to="." from="*"/>
          </output>
          <input message="inmsg"/>
        </operation>
        <operation name="Form Reject Response">
          <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">Copy Profile is undefined or Responder forced to
reject</assign>
            <assign to="Info">Unable to determine copy mode or FileInfo force
responder's rejection</assign>
            <assign to="deliveryNotification"
from="SwiftServerRequest/deliveryNotification/text()"/>
            <assign to="SnF" from="SwiftServerRequest/SnF/text()"/>
          </output>
          <input message="testing">
            <assign to="." from="*"/>
          </input>
        </operation>
      </sequence>
    </choice>
    <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>
    <choice name="IsThirdPartyForceRefusal">
      <select>
        <case ref="ForceRefusal" activity="InvokeForceRefusalProcess"/>
      </select>
      <operation name="InvokeForceRefusalProcess">
        <participant name="InvokeBusinessProcessService"/>
        <output message="Invoke_In">
          <assign to="." from="*"/>
          <assign to="INVOKE_MODE">ASYNC</assign>
          <assign to="WFD_NAME">SWIFTNet3rdPartyClientForceRefusal</assign>
        </output>
        <input message="Invoke_Out">
          <assign to="." from="*"/>
        </input>
      </operation>
    </choice>
    <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>
```

Parameters Passed From Business Process to Adapter

The following table contains the parameters passed from the business process to the SWIFTNet Server adapter:

Parameter	Description
messageID	Message identifier for the incoming message. Required.
interfaceMode	This is the SWIFTNet interface. Possible values are InterAct (default) or FileAct. Required.
deliveryNotification	Determines whether the server is handling a delivery notification. Valid values are True and False. Required.
SnF	Indicates whether you are using the store-and-forward method. Valid values are True (use store-and-forward) and False (do not use store-and-forward—this is the default). Required.

Parameter	Description
Status	The status of the message. Possible values are: ◆ Accepted ◆ Rejected ◆ Failed ◆ Duplicated Required.
Description	Description of the message. Optional. **Note:** Only necessary when there is an error message.
Information	Information about the message. Optional. **Note:** Only necessary when there is an error message.

Enabling SWIFTNet Document Tracking

You need to enable document tracking in the system business process you are using for the SWIFTNet Server adapter—handleSWIFTNetServerRequest (if you are not using store-and-forward processing) or handleSWIFTNetServerSnFRequest (if you are using store-and-forward processing)—so the system can track the document during the process. In the business process text editor, you can easily enable SWIFTNet document tracking in the application by selecting 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.

SWIFTNet Header Info Support

With SWIFTNet version 6.1, SWIFT introduced a new FileAct header field (HeaderInfo) to contain key summary information related to the file. The presence of Sw:HeaderInfo within the request is an indication to invoke the feature to validate the Sw:HeaderInfo. With SWIFTNet version 6.3, SWIFT will perform stronger central validation on the HeaderInfo fields for FileAct.

Once a service is activated for the validation, SWIFT checks the HeaderInfo contents (that is, presence, syntax, and semantic). SWIFT rejects files with HeaderInfo contents that either do not pass this validation, or that do not use the HeaderInfo field according to the rules defined for the service.

Header Info Support on the Client (the Application as Requestor)

As a requestor, the HeaderInfo is only allowed on the FileAct Put Request message. To specify the Header Info information, you must set the HeaderInfo parameter in the SWIFTNet Client Service accordingly. Please refer to *SWIFTNet Client Service* documentation for more details.

The application then validates the HeaderInfo if there is a matching **Request Type** profile in the SWIFTNet Service Profile before sending out the HeaderInfo. Therefore, when you specify HeaderInfo during sending, you must configure the **Request Type** profile in the SWIFTNet Service Profile. You can configure the

HeaderInfo as a mandatory or an optional parameter; however, if there is no matching profile, this request is forbidden.

Note: Please do not include <Sw:HeaderInfo> tag when specifying the value of HeaderInfo parameter. This <Sw:HeaderInfo> tag is automatically included during the process.

Header Info Support on the Server (the Application as Responder)

As a responder, the HeaderInfo is only allowed on the FileAct Get Response message. When the FileAct Get request is received, the responder checks to see if there is a HeaderInfo file located in the same directory as the download file. The HeaderInfo file must have the same name as the logical filename specified in the request except with an additional filename extension (.hdr). For example, if the logical filename is **payload.txt**, the HeaderInfo filename should be **payload.txt.hdr**. Prior to the Get request, the responder is responsible to provide both the download file and the HeaderInfo file in the correct directory.

When the FileAct Get request is received, this HeaderInfo is validated by the application using the SWIFTNet Service Profile. Depending on the Request Type profile in the SWIFTNet Service Profile, the HeaderInfo file can be mandatory, optional, or forbidden.

Note: Please do not include <Sw:HeaderInfo> tag when specifying the content of HeaderInfo file (*.hdr). This <Sw:HeaderInfo> tag is automatically included during the process.

SWIFTNet Service Profile

The HeaderInfo block is optional, except for those services that mandate it. If the HeaderInfo block is not used, it must not be present, and if it is used, it must be validated by the schema.

The SWIFTNet Service Profile enables you to easily port Service Profiles from one application instance to another. This function allows you to associate SWIFTNet Request Type with a Schema for Header Validation. You need to create the SWIFTNet Service Profile and associate the request type with the selected schema. This allows the application to validate the HeaderInfo when it is present in the request.

Note: The schema must be saved in application.

The Request Type parameter can accept a wildcard (*) to be used only at the end of the string. To determine which Service Profile to be used for a particular Request Type, the application uses a best-match policy. For example, if there are two Service Profile defined, for pain.* and pain.001.*, and the actual request type is pain.002.001, then the first one will be selected.

Two SWIFTNet Service Profiles are preloaded into application. The **pacs.*** and **pains.*** service profiles are associated with the Transaction Count schema and set to **Required for validation**. The Transaction Count and Payment Summary schemas are also preloaded into the application.

You can also import and export SWIFTNet Service Profiles from one application instance to another.

Creating a SWIFTNet Service Profile

To create a SWIFTNet service profile:

8. From the application **Deployment** menu, select **Adapter Utilities > SWIFTNet Service Profile**.
9. To the right of **Create new SWIFTNet service profile,** click **Go!**.

10. Complete the following parameters and click **Next**:

Parameter	Description
Request Type	Type the request type. A wildcard (*) is only allowed at the end of the string, for example for example, **pacs.*** or **pacs.001.***. Required.
Schema Name	Select the schema used to validate the header information for this request type. Required.
Validation Type	Select whether validation is mandatory or should only be used if header information is specified. Optional. Valid values are: ◆ Validates only if Header Information is specified (default) ◆ Validation of Header Information is required

11. Click **Finish** to save the service profile.

Searching for a SWIFTNet Request Type

To edit or delete a SWIFTNet request type, you must first locate the appropriate request type. You can locate a specific request type in two ways:

- ✦ Search for the request type by name.
- ✦ Select the request type from an alphabetical list.

Searching for the request type by name is more precise and provides fewer results. Searching from an alphabetical list will result in a list of all request type or all types beginning with a specified letter or digit.

Once you search for the request type, you can easily edit or delete it from the SWIFTNet Service Profile interface.

Searching for a Request Type by Name

To search for a request type by name:

1. From the application **Deployment** menu, select **Adapter Utilities > SWIFTNet Service Profile**.
2. In the Search section, type the name of the request type. Case does not matter and you can type part of a name.

   The application returns a list of matches unless no request type meet the criteria you specified.
3. When the list of matches is returned, click **edit** next to the request type you want to modify, or click **delete** next to the request type you want to remove.

Searching for a Request Type from a List

To select a request type from a list:

1. From the application **Deployment** menu, select **Adapter Utilities > SWIFTNet Service Profile**.
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!**

The application returns a list of matches unless no request type meet your criteria.

3. When the list of matches is returned, click **edit** next to the request type you want to modify, or click **delete** next to the request type you want to remove.

Exporting and Importing a SWIFTNet Service Profile

The application Import/Export feature enables you to save time and increase the accuracy of duplicating supported resources on different 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.