SWIFTNet Client Service

The SWIFTNet Client service is responsible for sending SWIFT InterAct or FileAct messages (both requests and responses) to SWIFTNet, which are initiated by the application. The SWIFTNet Client service enables you to use InterAct or FileAct messaging with a Store and Forward option. Additionally, the SWIFTNet Client service enables you to use either synchronous or asynchronous messaging for InterAct and either put or get messaging for FileAct.

Note: Each instance of the SWIFTNet Client service is configured for a pair of requestor/responder DNs and the SWIFTNet Client service name. This service may also be used by a third party to send authorization and refusal messages.

The following table provides an overview of the SWIFTNet Client service:

System Name	SWIFTNetClientService
Graphical Process Modeler (GPM) categories)	All Services.
Description	This service is responsible for sending SWIFT InterAct and FileAct messages (both requests and responses) to SWIFTNet, which are initiated by the application. The SWIFTNet Client service is also executed during system processing to create the SWIFTNet message header based on the configuration set in the CHIPS adapter. The request type is either chips.payment (if the transaction code is 10) or chips.message (for all transaction codes except 10). You do not need to specifically configure the SWIFTNet Client service for use with CHIPS.
Business usage	Use this service to send financial information based on SWIFT InterAct and FileAct messages to another participant in the SWIFTNet Central network. The business value of this service is inherent in utilizing the benefits of the SWIFTNet Central network to exchange financial messages.
Usage example	You wish to cancel a Customer Credit Transfer (MT192). The user uses the service to send out the Cancel Request, and wait for confirmation (MT 196). In this case, the request is sent out synchronously, with the service remaining open until the response is received.
Preconfigured?	Yes.
Requires third party files?	No third party files are required.
Platform availability	All supported application platforms.
Related services	This service works with the SWIFTNet Server adapter, the SWIFTNet HTTP Server adapter, the SWIFTNet MEFG Server, and the Command Line Adapter 2.
Application requirements	The SWIFTNet MEFG Server must be installed and configured in order to use this service. SSL can be implemented between the application and the SWIFTNet MEFG Server. You must also configure the SWIFTNet HTTP Server adapter.
Initiates business processes?	No.

Invocation	A user who has permission to perform this activity must execute the business process that invokes this service.
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	To test this adapter, run the SWIFTNet Client business process and verify that it completes successfully. Debug information for this service is located at: Operations > System > Logs > SWIFTNet

How the SWIFTNet Client Service Works

The SWIFTNet Client service prepares the request and sends it to the 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 Central network. The SWIFTNet Client service can operate in either synchronous or asynchronous mode for InterAct. In synchronous mode, the request is sent to the SWIFTNet Central network using the SwInt:Exchange primitive. In asynchronous mode, the request is sent to the SWIFTNet Central network using the SwInt:Send primitive.

In synchronous mode, the SWIFTNet MEFG Server 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 back to the application by the client application on the SWIFTNet MEFG Server, and the response payload is placed in the primary document.

In asynchronous mode, the SWIFTNet MEFG Server client application receives a response handle from the SAG/SNL instance. Using this response handle, the SWIFTNet MEFG Server client application periodically checks with SWIFTNet (using SwInt:Wait primitive) to determine if a response is available. Once a response is received, the response payload is placed in the primary document.

With release SWIFT 6.1, the use of a messaging interface that uses input channels is optional but is recommended. SWIFT plans to make it mandatory in a future release. Currently, the use of input channels is available for SWIFTNet InterAct Store and Forward (SnF) only.

If you configure the Input Channel in the SWIFTNet Server adapter, the SWIFTNet MEFG Server opens the Input Channel automatically during startup (when the SWIFTNet Server adapter is enabled). This Input Channel remains open until the SWIFTNet MEFG Server is shut down (when the SWIFTNet Server adapter is disabled). During this time, you have the option to send message using the input channel or without the input channel if you indicate this using the Input Channel parameter in the SWIFTNet Client service.

When SWIFTNet Copy is used, the Copied Request is sent to a third party in one of two modes: T-Copy (third party copy is for information only) or Y-Copy (third party needs to do authorization). When SWIFTNet Copy is used, the message/file copy is queued for delivery to the configured third party as soon as the third party is ready to receive. The third party has to acknowledge the receipt of the copy like any other message or file delivered from a queue. When the mode is Y-copy, then the third party must authorize or refuse the message or file, which requires a system message to be sent to SWIFT. When authorizing the message or file, the authorization may contain information destined to the sender and information destined to the receiver of the message or file.

When using SWIFT 6.1, the application may act as a third party. If the Copy Mode is Y-Copy, the application sends an authorization message, which is like sending an Interact store-and-forward request. The SWIFTNet Client service is used, but you must set the you must set the **thirdPartyAuth** parameter to TRUE, and provide the authorization decision (either Authorised or Refused) for the **AuthDecision** parameter. Additionally, the third party may provides information destined to the sender or receiver of the message or file.

Implementing the SWIFTNet Client Service

To implement the SWIFTNet Client service, complete the following tasks:

1. Create a configuration of the SWIFTNet Client service. See *Managing Services and Services*. For information about the fields specific to this service, see *Configuring the SWIFTNet Client Service* on page 3.

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

Note: If you create a new configuration of the SWIFTNet Client service, you must also create a new business process or edit a copy of the appropriate predefined business process, SWIFTNetClient.bp or SWIFTNetClientFA.bp, to update it to use your service configuration. You do not need to create an instance of the SWIFTNet Client service for every Requestor or Responder DN; you can simply reuse the SWIFTNet Client service instance and pass the parameters that differ from the sample service through the business process.

2. Specify field settings for the service configuration in the application Admin Console and in the GPM as necessary. See *Configuring the SWIFTNet Client Service* on page 3.

Note: When you create the configuration, you will configure it differently depending on whether you are using InterAct or FileAct messaging. Either can be used with or without the store-and-forward option.

Configuring the SWIFTNet Client Service

1. Select **Deployment > Services > Configuration**.
2. Search for SWIFTNet Client service or select it from the list and click **Go!**.
3. Click **Edit**.
4. Specify field settings in the Admin Console or Business Process (*Creating or Setting Up a Service Configuration in the Admin Console or Business Process* on page 4), or the GPM (*Setting Up the Service in the GPM* on page 8).

Note: Each instance of the SWIFTNet Client service is configured for a pair of requestor/responder DNs and the SWIFTNet Client service name.

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

Creating or Setting Up a Service Configuration in the Admin Console or Business Process

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

Field	Description
Name	Unique and meaningful name for the service configuration. Required.
Description	Meaningful description for the service configuration, for reference purposes. Required.
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:** See *Managing Services and Services*.
SWIFTNet Interface	SWIFTNet message type. Valid values are **InterAct** or **FileAct**. Required.
Store and Forward	Indicates if the file transfer is done using the store-and-forward method. Valid values are True (use Store-and-Forward) and False (default—do not use Store-and-Forward). Required. BPML element value is **SnF**.
SWIFTNet Operation	The SWIFTNet operation to send an InterAct or FileAct message. Possible values are: ◆ Synchronous (default)—InterAct ◆ Asynchronous—InterAct ◆ Put—FileAct ◆ Get—FileAct Required. When SWIFTNet Operation is FileAct, you must select either Put or Get. If you do not select an operation, the service uses Synchronous as the default value. BPML element value is **sync** (default) or **async** for InterAct, or **Put** or **Get** for FileAct.
Requestor DN	Distinguished name of the requestor. Required. BPML element value is **requestorDN**. **Note:** This DN must be registered with the SAG instance using SWIFTNet Alliance Webstation.
Responder DN	Distinguished name of the responder. Required. BPML element value is **responderDN**. **Note:** This DN must be registered with the SAG instance using SWIFTNet Alliance Webstation.

Field	Description
Service Name	Name of the service to which both SWIFT correspondents have subscribed. Required. BPML element value is **serviceName**. **Note:** This must be a SWIFTNet service to which you are subscribed.
Authoriser DN	The distinguished name of the authorizing party. Optional.
This service allows Third Party Copy	Whether this service uses T-Copy or Y-Copy (check your service agreement with SWIFT). BPML element value is **thirdPartyCopy**. Valid values are TRUE or FALSE. This parameter is displayed only if you selected **File Act** and **True** for **Store and Forward** on SWIFTNet Client Service Interface page. **Note:** If the Copy Mode is Y-Copy, the application sends an authorization message, which is like sending an Interact store-and-forward request. The SWIFTNet Client service is used, but you must set the **This service allows Third Party Copy** parameter to TRUE, and provide the authorization decision (either Authorised or Refused) for the AuthDecision parameter.
Request for Third Party Copy	Whether you are requesting third party copy. When the Copy feature is defined as Optional in the service agreement, you can choose whether you want the Third Party Copy to occur. BPML element value is **copyIndicator**. Valid values are TRUE or FALSE. Displayed only if you select True for **This service allows Third Party Copy**. **Note:** This parameter is displayed only if you selected **True** for **This service allows Third Party Copy**.
Request for Notification from Third Party	In T-Copy mode, this setting is not applicable, the value should always be set to FALSE. In Y-Copy mode, when the **Authorisation Notification Indicator** feature is available and defined as Optional in the service agreement, you can choose whether you want to receive the Authorisation Notification messages. BPML element value is **authNotifIndicator**. Valid values are TRUE or FALSE. Displayed only if you selected **True** for **This service allows Third Party Copy**. **Note:** This parameter is displayed only if you selected **True** for **This service allows Third Party Copy**.
Request Type	Request type supported by the message exchange. Optional for InterAct and required for FileAct in SWIFTNet 6.0. BPML element value is **requestType**. **Note:** In SWIFTNet 6.0 FileAct the format convention is as follows: `<business_area>.<type_of_syntax>.<detailed_syntax_and_format>` This format starts with a four-characters business area code, followed by a period (dot), followed by a three-character code that designates the type of syntax (which can be <nnn> , FIN, or xxx), followed by another period (dot), and then followed by a more detailed indication of syntax and format.
Request Reference	User reference of the request. Optional. BPML element value is **requestReference**.
Non Repudiation Required	Indicates whether non-repudiation is required. Possible values are True (when enabled this means that trading partners cannot deny that they sent a request) or False (default—when enabled this indicates that non-repudiation is not required). Optional. BPML element value is **nonRepudiation**.

Field	Description
End-to-End Signature Required	Whether an end-to-end signature is required. Valid values are False (default) and True. Optional. **Note:** You can use an end-to-end signature regardless of whether you are using non-repudiation (for example, for SWIFT SCORE messages).
Number of Retries	Number of retries to connect to SAG. Default value is 3. Optional. BPML element value is **numOfRetries**.
Retry Delay (in seconds)	Time that will elapse before the next retry. Default value is 60 (seconds). Optional. BPML element value is **secInRetryDelay**.
Trace	Trace for logging purposes in the SWIFTNet MEFG Server. Valid values are True and False (default). Required. BPML element value is **trace**.
Use Signature List	Whether to use a signature list. This enables you to select your own signatures. If you do not use a signature list then normal Crypto is used. Valid values are False and True. Required. **Note:** This parameter is displayed only if you selected **True** for **End-to-End Signature Required**.
Return Signature List	Whether to return a signature list. Valid values are False and True. Required. If you want a signature list returned, the SWIFTNet MEFG Server receives the requestor's own signature in the response message. This returned signature will be extracted and saved as a separate message. This message is stored in the database and is made available for Correlation search. **Note:** This parameter is displayed only if you selected **True** for **End-to-End Signature Required**.
Use RND	Whether to use RND (digest reference values that terminate on “and RND”). Valid values are False (default) and True. Required. **Note:** This parameter is displayed only if you selected **True** for **End-to-End Signature Required**.
Delivery Notification (Del. Notifn)	Indicates that the sender asked the receiver to send a delivery notification. Possible values are True or False (default). Optional. BPML element value is **deliveryNotification**. **Note:** This parameter is only displayed when you select **True** for Store and Forward or are performing a FileAct Put. If you are performing a Put operation, you can request the responder to send you a delivery notification and specify a different Delivery Notification DN and Request Type of Delivery Notification, if desired. If you are performing a Get operation, the responder can request Delivery Notification from the requestor after receiving the file. That setting for delivery notification is configured through the SWIFTNet Server adapter.
Request Type of Delivery Notification	Used to request a specific delivery notification message from the remote receiving server application when it returns the delivery notification (when Delivery Notification is set to True). Optional. BPML element value is **requestTypeDelNotifn**. **Note:** This parameter is only displayed when you select **True** for Store and Forward or a FileAct Put.

Field	Description
Message Priority	Indicates priority handling in the queue for store-and-forward only. Valid values are Normal (default) and Urgent. Optional. BPML element value is **messagePriority**. **Note:** This value is used as a selection criterion when delivering messages from a queue, and in SWIFTNet FileAct to influence the pace of the FileAct flow.
Use Input Channel	Whether to use the input channel. Valid values are False (default) and True. Required. This parameter is displayed only if you selected **True** for **Store and Forward** and **InterAct** for **SWIFTNet interface**. **Note:** Used for InterAct store-and-forward only. Select **True** if you are using an input channel. 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.
MEFG SWIFTNet IP	The IP address for the SWIFTNet MEFG Server. Required.
MEFG SWIFTNet Port	The port for the SWIFTNet MEFG Server. Default is NULL. Optional.
Response Timeout	The timeout interval (in seconds) in which a response must be received or the message operation fails. Optional. Default is 60 seconds.
Use SSL	Whether to enable Secure Socket Layer (SSL) over HTTP communication between the application and the SWIFTNet MEFG Server. Valid values are None and Must. **Note:** Regardless of the value you select for **Use SSL**, you must also update the business processes associated with the SWIFTNet Client service. See *Upgrading the SWIFTNetClient Business Process to Use the Integrated SWIFTNet Client Service* on page 22 for more information.
Cipher Strength	Indicates the strength of the cipher. Possible values are ALL (default), WEAK, and STRONG. Optional.
CA Certificate	The CA certificate of the SWIFTNet MEFG Server. **Note:** This is the public key certificate that must be configured to set up the outbound SSL channel. This page is only displayed if you set **Use SSL** to **Must**. **Note:** The SWIFTNet Client service Configuration page allows you to select the same CA Certificate for SSL processing a second time, and continues to allow additional selections of the same certificate in subsequent edits. If you have already selected a CA Certificate once for a configuration of the SWIFTNet Client service, do not select the same CA Certificate again, as this will result in an error when you execute the relevant business process.
Switch to SnF mode when real-time transmission failed	Whether to switch to store-and-forward mode when real-time transmission fails. Select True if you want to switch to Store and Forward mode when the real-time transmission (InterAct and FileAct Put) has failed. Valid values are True and False.

Setting Up the Service in the GPM

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

Parameter	Description
deliveryNotification	Indicates that the sender asked the receiver to send a delivery notification. Possible values are TRUE or FALSE. Optional. **Note:** This parameter is only displayed when you select **True** for SnF or are performing a FileAct Put. If you are performing a Put operation, you can request the responder to send you a delivery notification and specify a different Delivery Notification DN and Request Type of Delivery Notification, if desired. If you are performing a Get operation, the responder can request Delivery Notification from the requestor after receiving the file. That setting for delivery notification is configured through the SWIFTNet Server adapter.
deliverynotification DN	Distinguished name of the Responder of the delivery notification. Optional.
deliverynotificationRT	Request type of the delivery notification. This is used for a FileAct Get. Required.
fileDesc	User description about the file transfer. Only for FileAct Put. Optional.
fileInfo	Specify whether the file will be compressed or not. Only for FileAct Put. Optional. In SWIFTNet 6.0 FileAct, the format convention is as follows: `SwCompression=<value>` Valid values are SwCompression=None (default) or SwCompression=ZIP. **Note:** If you specify to use compression, you must have compressed the file before sending it to the SWIFTNet Server adapter.
interfaceMode	SWIFTNet message type. Valid values are InterAct or FileAct. The default value is InterAct. Required.
logicalFilename	This name is communicated to the server application. By default, this name is the physical name without the file path. Optional. Only for FileAct. For a FileAct Put, this is the logical name of the file to be retrieved based on the <reception_dir>/<responder_dn>/<requestor_dn>. For a FileAct Get, this is the logical name of the file to send based on the <download_dir>/<responser_dn>/<requestor_dn>.
messagePriority	Indicates priority handling in the queue for store-and-forward only. Optional. **Note:** This value is used as a selection criterion when delivering messages from a queue, and in SWIFTNet FileAct to influence the pace of the FileAct flow.
nonRepudiation	Indicates whether non-repudiation is required. Possible values are TRUE (when enabled, trading partners cannot deny that they sent a request) or FALSE (default, indicating that non-repudiation is not required). Optional.
numOfRetries	Number of retries to connect to SAG. Default value is 3. Optional.
physicalFilename	Optional. Only for FileAct. For a FileAct Put, this is the full path and the physical name of the file to send. For a FileAct Get, this is the full path and the physical name of the file to save after the Get is completed.

Parameter	Description
possibleDuplicate	Indicates whether to include a trailer specifying that this message may be a duplicate. 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. Possible values are TRUE and FALSE (default). Optional.
requestorDN	Distinguished name of the requestor. Required. **Note:** This DN must be registered with the SAG instance using SWIFTNet Alliance Webstation.
requestReference	User reference of the request. Optional.
requestType	Request type supported by the message exchange. Optional for InterAct and required for FileAct in SWIFTNet 6.0. **Note:** In SWIFTNet 6.0 FileAct the format convention is as follows: `<business_area>.<type_of_syntax>.<detailed_syntax_and_format>` This format starts with a four-characters business area code, followed by a period (dot), followed by a three-character code that designates the type of syntax (which can be <nnn> , FIN, or xxx), followed by another period (dot), and then followed by a more detailed indication of syntax and format
requestTypeDelNotifn	Used to request a specific delivery notification message from the remote receiving server application when it returns the delivery notification (when Non Repudiation required and/or Delivery Notification are set to TRUE). Optional
responderDN	Distinguished name of the responder. Required. **Note:** This DN must be registered with the SAG instance using SWIFTNet Alliance Webstation.
secInRetryDelay	Number of delays before the next retry. Default value is 60 (seconds). Optional.
serviceName	Name of the service to which both SWIFT correspondents have subscribed. Required. This must be a SWIFTNet service to which you have already subscribed.
SnF	Indicates if the file transfer is done using the store-and-forward method. Valid values are True (use Store-and-Forward) and False (default—do not use Store-and-Forward). Required.
swiftOp	The SWIFTNet operation to send an InterAct or FileAct message Possible values are: ◆ sync (default)—InterAct ◆ async—InterAct ◆ put—FileAct ◆ get—FileAct Required.
trace	Trace for logging purposes in the SWIFTNet MEFG Server. Possible values are 0 (no logging; this is the default) or 4 (logging is enabled). Optional.
transferDesc	User description about the transfer. Only for FileAct. Optional.

Parameter	Description
transferInfo	User information about the transfer. Only for FileAct. Optional.
switchToSnF	Indicates whether you want to switch to store-and-forward mode if a real-time transmission (InterAct or a FileAct Put) has failed. Possible values are True or False (default). Required.
SnFServiceName	The name of the store-and-forward service. Required when Switch to SnF mode when real-time transmission failed is set to True.
HTTPClientAdapter	HTTP Client Adapter instance that is used to communicate with the SWIFTNet MEFG Server. Optional. Default value is SWIFTNetHTTPClientAdapter.
MEFGServerHost	The IP address of the SWIFTNet MEFG Server. Required.
MEFGServerPort	The port of the SWIFTNet MEFG Server. Optional. Default value is 80.
MEFGServerResponse Timeout	Timeout period (in seconds) for the SWIFTNet MEFG Server to respond. Optional. Default value is 60.
UseSSL	Flag to indicate whether to secure communication between the application and the SWIFTNet MEFG Server with SSL. Possible values are TRUE or FALSE (default). Optional.
CipherStrength	The level of encryption to be applied on the data channel. Possible values are All (default), Weak, or Strong. Optional.
CACertId	The public key certificates for the SWIFTNet MEFG Server. Required if SSL is set to TRUE.
sign	Whether an end-to-end signature is required.
useSignatureList	Whether to use a signature list. This enables you to select your own signatures. If you do not use a signature list then normal Crypto is used.
returnSignatureList	Whether to return a signature list.
useRND	Whether to use RND (digest reference values that terminate on “and RND”). Valid values are False (default) and True.
useInputChannel	Whether to use the input channel.
messageID	The message identifier of the payload. This is used only when **Possible Duplicate** is set to **True**.
renewDN	The distinguished name of the user. This is used for the renewal of Security Context.
authoriserDN	The distinguished name of the authorizing party.
thirdPartyCopy	Flag to indicate whether this service should use T-Copy or Y-Copy. Only available for FileAct SnF. Valid values are TRUE or FALSE.
copyIndicator	When the Third Party Copy feature is defined as Optional in the service agreement, you can choose whether you want Third Party Copy to occur. Valid values are TRUE or FALSE.

Parameter	Description
authNotifIndicator	In T-Copy mode this setting is not applicable and the value should always be set to FALSE. In Y-Copy mode, when the **Authorisation Notification Indicator** feature is available and defined as Optional in the service agreement, you can choose whether you want to receive back the Authorisation Notification messages. Valid values are TRUE or FALSE.
HeaderInfo	Your Enhanced Header Info. Since Enhanced Header Info is usually an XML structure, you should specify it as CDATA.
thirdPartyAuth	Flag to indicate that the application is acting as Third Party who are going to send notification message (in Y-Copy mode). Valid values are TRUE or FALSE. This parameter should be used together with the required **AuthDecision** and **MessageName** parameters, and optional **ToSndrInfo**, **ToRcvrInfo** and **RefuseReason** parameters.
AuthDecision	Specify the third party decision here. Valid values are Authorised or Refused. Use this parameter with **thirdPartyAuth** parameter.
MessageName	Specify the name of the HeaderInfo message as inserted into the mailbox. The format is ThirdParty_[CopySnFRef]. When you use Mailbox Extract service to extract the HeaderInfo message from mailbox, by default the name is available in Process Data.
ToSndrInfo	If you are the third party and decide to Authorised a request or file notifications, this parameter enables you to specify “Third Party to Sender Information.” The information can be any structure (plain text or XML). If you use XML structure, this parameter should be CDATA.
ToRcvrInfo	If you are the third party and decide to Authorised a request or file notifications, this parameter enables you to specify Third Party to Receiver Information. The information can be any structure (plain text or XML). If you use XML structure, this parameter should be CDATA.
RefuseReason	If you are the third party and you refuse a request or file notifications, you can specify Refusal Reason in this parameter. The information can be any structure (plain text or XML). If you use XML structure, this parameter should be CDATA.

Business Process Example

To construct a message you need to perform the following tasks:

- Create a configuration of the SWIFTNet Client service.
- Edit the SWIFTNetClient business process (or create a new business process) in the following manner:
  - Match the name of the business process that you create or modify.
  - If necessary, modify the SWIFTNet MEFG Server IP and port to point to your installation of the SWIFTNet MEFG Server.
  - Configure the business process for the Requestor DN/Responder DN pair and the SWIFTNet service name.
  - Specify the request type and request reference for use in SWIFTNet.
  - If required, select non-repudiation and possible duplicate (which enables the resending of the file in case of an error in transmission) parameters.

- Specify the number of retries to the SAG connection and the retry interval.
- Enable Document Tracking for AFT Tracking.

Note: You do not need to create an instance of the SWIFTNet Client service for every requestor or responder DN; you can reuse the SWIFTNet Client service instance and pass in the requestorDN, responderDN, and any other parameters that differ from the configuration of the sample service through the SWIFTNetClient business process.

This is the BPML for the example business process:

```
<operation>
<participant name="SWIFTNetClientService"/>
<output message="handleClientRequest">
<assign to="." from="*"/>
<assign to="swiftOp">async</assign>
</output>
<input message="testing">
          <assign to="." from="*"/>
</input>
</operation>
```

This is the complete BPML to execute the SWIFTNet Client service:

Note: The **bold** lines indicate information that you need to modify to match the business process you are using.

```
<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="*"></assign>
      </output>
      <input message="inmsg">
        <assign to="." from="*"></assign>
      </input>
    </operation>

    <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=swiftbic,o=swift</assign>
        <assign to="responderDN">o=swiftbic,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>
      </output>
      <input message="testing">
        <assign to="." from="*"></assign>
      </input>
```

```
    </operation>

  </sequence>
</process>
```

This is the complete BPML to execute the SWIFTNet Client service for FileAct for a Put:

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

    <operation>
      <participant name="SWIFTNetClientService"/>
      <output message="handleClientRequest">
        <assign to="." from="*"></assign>
        <assign
to="physicalFilename">/local/share/measle/swiftdata/payload.txt</assign>
        <assign to="logicalFilename">payload.txt</assign>
        <assign to="transferInfo">payload</assign>
        <assign to="transferDesc">payload></assign>
        <assign to="fileDesc">payload</assign>
        <assign to="interfaceMode">fileact</assign>
        <assign to="swiftOp">put</assign>
        <assign to="requestorDN">o=swiftbic,o=swift</assign>
        <assign to="responderDN">o=swiftbic,o=swift</assign>
        <assign to="serviceName">swift.generic.fa!x</assign>
        <assign to="requestType">Type.GIS.Server1</assign>
        <assign to="SnF">FALSE</assign>
        <assign to="nonRepudiation">FALSE</assign>
        <assign to="possibleDuplicate">FALSE</assign>
        <assign to="deliveryNotification">TRUE</assign>
      </output>
      <input message="testing">
        <assign to="." from="*"></assign>
      </input>
    </operation>

  </sequence>
</process>
```

This is the complete BPML to execute the SWIFTNet Client service for FileAct for a Get:

```
<process name="SWIFTNet-FA-Get-NonSnF-DN">
  <sequence name="SWIFTNetClientService">
    <operation name="set user token">
      <participant name="SetUserToken"/>
      <output message="SetUserTokenMessage">
        <assign to="USER_TOKEN">admin</assign>
```

```
        <assign to="." from="*"></assign>
      </output>
      <input message="inmsg">
        <assign to="." from="*"></assign>
      </input>
    </operation>

    <operation>
      <participant name="SWIFTNetClientService"/>
      <output message="handleClientRequest">
        <assign to="." from="*"></assign>
        <assign
to="physicalFilename">/local/share/measle/swiftdata/payload-receive.txt</assign>
        <assign to="logicalFilename">payload.txt</assign>
        <assign to="interfaceMode">fileact</assign>
        <assign to="swiftOp">get</assign>
        <assign to="requestorDN">o=swiftbic,o=swift</assign>
        <assign to="responderDN">o=swiftbic,o=swift</assign>
        <assign to="serviceName">swift.generic.fa!x</assign>
        <assign to="SnF">FALSE</assign>
        <assign to="nonRepudiation">FALSE</assign>
        <assign to="possibleDuplicate">FALSE</assign>
        <assign to="deliveryNotification">TRUE</assign>
      </output>
      <input message="testing">
        <assign to="." from="*"></assign>
      </input>
    </operation>

  </sequence>
</process>
```

This is a sample business process for a third party to send an authorised notification message:

```
<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>
    <operation>
      <participant name="SWIFTNetClientService"/>
      <output message="handleClientRequest">
        <assign to="." from="*"/>
        <assign to="thirdPartyAuth">TRUE</assign>
        <assign to="AuthDecision">Authorised</assign>
        <assign to="MessageName">ThirdParty_snp892349710118</assign>
        <assign to="ToSndrInfo">Plain text example</assign>
        <assign to="ToRcvrInfo"><![CDATA[<info><abc>XML
example</abc></info>]]></assign>
      </output>
```

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

This is a sample business process for a third party Third Party to send a refused notification message.

```
<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>
    <operation>
      <participant name="SWIFTNetClientService"/>
      <output message="handleClientRequest">
        <assign to="." from="*"/>
        <assign to="thirdPartyAuth">TRUE</assign>
        <assign to="AuthDecision">Refused</assign>
        <assign to="MessageName">ThirdParty_snp892349710118</assign>
        <assign to="RefuseReason">Plain text example</assign>
      </output>
      <input message="testing">
        <assign to="." from="*"/>
      </input>
    </operation>
  </sequence>
</process>
```

This is a sample business process to send a FileAct store-and-forward with Header Info:

```
<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>
    <operation>
      <participant name="SWIFTNetClientService"/>
      <output message="handleClientRequest">
        <assign to="." from="*"/>
        <assign to="interfaceMode">fileact</assign>
        <assign to="swiftOp">put</assign>
        <assign to="SnF">TRUE</assign>
```

```
        <assign to="requestorDN">o=swiftbic,o=swift</assign>
        <assign to="responderDN">o=swiftbic,o=swift</assign>
        <assign to="serviceName">swift.generic.fa!x</assign>
        <assign to="requestType">pain.001.001.01</assign>
        <assign
to="physicalFilename">/local/share/measle/swiftdata/payload.txt</assign>
        <assign to="logicalFilename">payload.txt</assign>
        <assign to="transferInfo">Date=29082008</assign>
        <assign to="transferDesc">transfer desc</assign>
        <assign to="fileInfo">SwCompression=None</assign>
        <assign to="fileDesc">file desc</assign>
        <assign to="HeaderInfo"><![CDATA[<ApplSpcfc
xmlns="urn:swift:xsd:ApplSpcfc.TxsCntr.01"><TxsCntr><TtlNbOfTxs>5</TtlNbOfTxs></TxsC
ntr></ApplSpcfc>]]></assign>
        <assign to="nonRepudiation">FALSE</assign>
        <assign to="possibleDuplicate">FALSE</assign>
        <assign to="deliveryNotification">FALSE</assign>
      </output>
      <input message="testing">
        <assign to="." from="*"/>
      </input>
    </operation>
  </sequence>
</process>
```

This is the complete business process to open the input channel:

```
<process name="SWIFTNetOpenInputChannel">
   <sequence name="SWIFTNetOpenInputChannel">
      <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 Open request -->
       <operation name="Service">
         <participant name="SWIFTNetClientService"/>
         <output message="openInputChannelRequest">
            <assign to="." from="*"/>
         </output>
         <input message="inmsg">
            <assign to="." from="*"/>
         </input>
       </operation>
  </sequence>
</process>
```

This is the complete business process to close the input channel:

```
<process name="SWIFTNetCloseInputChannel">
   <sequence name="SWIFTNetCloseInputChannel">
      <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 Close request -->
        <operation name="Service">
            <participant name="SWIFTNetClientService"/>
            <output message="closeInputChannelRequest">
                <assign to="." from="*"/>
            </output>
            <input message="inmsg">
                <assign to="." from="*"/>
            </input>
        </operation>
    </sequence>
</process>
```

This is the complete business process to create the input channel:

```
<process name="SWIFTNetCreateInputChannel">
    <sequence name="SWIFTNetCreateInputChannel">
        <operation name="set user token">
            <participant name="SetUserToken"/>
            <output message="SetUserTokenMessage">
                <assign to="USER_TOKEN">admin</assign>
                <assign to="." from="*"></assign>
            </output>
            <input message="inmsg">
                <assign to="." from="*"></assign>
            </input>
        </operation>
        <!-- build Create request -->
        <operation>
            <participant name="SWIFTNetClientService"/>
            <output message="createInputChannelRequest">
                <assign to="." from="*"></assign>
                <assign to="authoriserDN">Put a value here</assign>
                <assign to="inputChannelName">Put a value here</assign>
            </output>
            <input message="inmsg">
                <assign to="." from="*"></assign>
            </input>
        </operation>
    </sequence>
</process>
```

This is the complete business process to delete the input channel:

```
<process name="SWIFTNetDeleteInputChannel">
    <sequence name="SWIFTNetDeleteInputChannel">
        <operation name="set user token">
            <participant name="SetUserToken"/>
            <output message="SetUserTokenMessage">
                <assign to="USER_TOKEN">admin</assign>
```

```
            <assign to="." from="*"></assign>
         </output>
         <input message="inmsg">
            <assign to="." from="*"></assign>
         </input>
      </operation>
      <!-- build Delete request -->
      <!--  W A R N I N G   N O T E  -->
      <!-- Once deleted, the input channel cannot be re-created or used anymore -->
      <operation>
         <participant name="SWIFTNetClientService"/>
         <output message="deleteInputChannelRequest">
            <assign to="." from="*"></assign>
            <assign to="authoriserDN">Put a value here</assign>
            <assign to="inputChannelName">Put a value here</assign>
         </output>
         <input message="inmsg">
            <assign to="." from="*"></assign>
         </input>
      </operation>
   </sequence>
</process>
```

This is the complete business process to renew the Security Context:

```
<process name="SWIFTNetClientRenewSecContext">
   <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="renewSecurityContext">
            <assign to="renewDN">o=swiftbic,o=swift
</assign>
            <assign to="." from="*"/>
         </output>
         <input message="testing">
            <assign to="." from="*"/>
         </input>
      </operation>
   </sequence>
</process>
```

Parameters Passed From Business Process to Service

The following table contains the parameters passed from the business process to the SWIFTNet Client service:

Parameter	Description
swiftOp	The SWIFTNet operation to send an InterAct or FileAct message Possible values are: ◆ sync (default)—InterAct ◆ async—InterAct ◆ put—FileAct ◆ get—FileAct Required.
trace	Trace for logging purposes in the SWIFTNet MEFG Server. Possible values are 0 (no logging; this is the default) or 4 (logging is enabled). Optional.
numOfRetries	Number of retries to connect to SAG. Default value is 3. Optional.
secInRetryDelay	Time that wil elapse before the next retry. Default value is 60 (seconds). Optional.
requestorDN	Distinguished name of the requestor. Required. **Note:** This DN must be registered with the SAG instance using SWIFTNet Alliance Webstation.
responderDN	Distinguished name of the responder. Required. **Note:** This DN must be registered with the SAG instance using SWIFTNet Alliance Webstation.
serviceName	Name of the service to which both SWIFT correspondents have subscribed. Required. **Note:** This must be SWIFTNet service to which you are subscribed.
requestType	Request type supported by the message exchange. Optional.
requestReference	User reference of the request. Optional.
SnF	Indicates if the file transfer is done using the store-and-forward method. Valid values are True (use Store-and-Forward) and False (default—do not use Store-and-Forward). Required.
physicalFilename	Optional. Only for FileAct. For a FileAct Put, this is the full path and the physical name of the file to send. For a FileAct Get, this is the full path and the physical name of the file to save after the Get is completed.
logicalFilename	This name is communicated to the server application. By default, this name is the physical name without the file path. Optional. Only for FileAct. For a FileAct Put, this is the logical name of the file to be saved based on the <reception_dir>/<responder_dn>/<requestor_dn>. For a FileAct Get, this is the logical name of the file to send based on the <download_dir>/<responser_dn>/<requestor_dn>.
fileInfo	User information about the file transfer. Only for FileAct. Optional.

Parameter	Description
fileDesc	User description about the file transfer. Only for FileAct. Optional.
transferInfo	User information about the transfer. Only for FileAct. Optional.
transferDesc	User description about the transfer. Only for FileAct. Optional.
possibleDuplicate	Indicates whether to include a trailer specifying that this message may be a duplicate. 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. Possible values are TRUE and FALSE (default). Optional.
messageID	Message identifier for resending a message if Possible Duplicate is set to TRUE. Optional.
deliveryNotification	Indicates that the sender asked the receiver to send a delivery notification. Possible values are TRUE or FALSE. Optional. **Note:** This parameter is only displayed when you select **True** for SnF or are performing a FileAct Put. If you are performing a Put operation, you can request the responder to send you a delivery notification and specify a different Delivery Notification DN and Request Type of Delivery Notification, if desired. If you are performing a Get operation, the responder can request Delivery Notification from the requestor after receiving the file. That setting for delivery notification is configured through the SWIFTNet Server adapter.
requestTypeDelNotifn	Used to request a specific delivery notification message from the remote receiving server application when it returns the delivery notification (when Non Repudiation required and/or Delivery Notification are set to TRUE). Optional
messagePriority	Indicates priority handling in the queue for store-and-forward only. Optional. **Note:** This value is used as a selection criterion when delivering messages from a queue, and in SWIFTNet FileAct to influence the pace of the FileAct flow.
nonRepudiation	Indicates whether non-repudiation is required. Possible values are TRUE (when enabled, trading partners cannot deny that they sent a request) or FALSE (default, indicating that non-repudiation is not required). Optional.
HeaderInfo	The Enhanced Header information. Since Enhanced Header Info is usually an XML structure, you should specify it as CDATA.
thirdPartyAuth	Flag to indicate that the application is acting as a Third Party that will send a notification message (in Y-Copy mode). Valid values are TRUE or FALSE. This parameter should be used together with the required AuthDecision and MessageName parameters, and optional ToSndrInfo, ToRcvrInfo and RefuseReason parameters.
AuthDecision	The third party decision. Valid values are Authorised or Refused. Use this parameter with the thirdPartyAuth parameter.
MessageName	The name of the HeaderInfo message, as inserted into the mailbox. The format is ThirdParty_[CopySnFRef]. When you use Mailbox Extract service to extract the HeaderInfo message from mailbox, by default the name is available in Process Data.

Parameter	Description
ToSndrInfo	If you are the third party and decide to authorize a request or file notification, this parameter enables you to specify Third Party to Sender Information. The information can be in any format (plain text or XML). If you use XML format, this parameter should be CDATA.
ToRcvrInfo	If you are the third party and decide to authorize a request or file notification, this parameter enables you to specify Third Party to Receiver Information. The information can be any format (plain text or XML). If you use XML format, this parameter should be CDATA.
RefuseReason	If you are the third party and you refuse a request or file notification, you can specify a Refusal Reason in this parameter. The information can be any format (plain text or XML). If you use XML format, this parameter should be CDATA.
sign	Whether an end-to-end signature is required.
useSignatureList	Whether to use a signature list. This enables you to select your own signatures. If you do not use a signature list then normal Crypto is used.
returnSignatureList	Whether to return a signature list.
useRND	Whether to use RND (digest reference values that terminate on “and RND”). Valid values are False (default) and True.
renewDN	The distinguished name of the user. This is used for the renewal of Security Context.
MEFGServerHost	The IP address of the SWIFTNet MEFG Server. Required.
MEFGServerPort	The port of the SWIFTNet MEFG Server. Optional. Default value is NULL.
MEFGServerResponse Timeout	Timeout period (in seconds) for the SWIFTNet MEFG Server to respond. Optional. Default value is 60.
UseSSL	Flag to indicate whether to secure communication between the application and the SWIFTNet MEFG Server with SSL. Possible values are TRUE or FALSE (default). Optional.
CACertId	The public key certificates for the SWIFTNet MEFG Server. Required if SSL is set to TRUE.
CipherStrength	The level of encryption to be applied on the data channel. Possible values are All (default), Weak, or Strong. Optional.
useInputChannel	Whether to use the input channel.
forceOpen	Whether to force the channel open.
switchToSnF	Whether to switch to store-and-forward mode when real-time transmission fails. Select True if you want to switch to Store and Forward mode when the real-time transmission (InterAct and FileAct Put) has failed. Valid values are True and False.
SnFServiceName	The name of the store-and-forward service. Required when Switch to SnF mode when real-time transmission failed is set to True.
HTTPClientAdapter	HTTP Client Adapter instance that is used to communicate with the SWIFTNet MEFG Server. Optional. Default value is SWIFTNetHTTPClientAdapter.

Upgrading the SWIFTNetClient Business Process to Use the Integrated SWIFTNet Client Service

Now that the SWIFTNet Client service has been enhanced to support SSL, the SWIFTNet Client service has also been improved by integrating all the outbound services internally. To use the SWIFTNet Client service, you must upgrade the SWIFTNetClient business process. The upgraded BPML differs based on whether you are using InterAct or FileAct.

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 *Using SWIFTNet* for more information).

Upgrading the SWIFTNetClient Business Process for InterAct

If you are using InterAct, this is the complete BPML to execute the SWIFTNet Client service for InterAct:

```
<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>
```

Upgrading the SWIFTNetClient Business Process for FileAct

If you are using FileAct, this is the complete BPML to execute the SWIFTNet Client service for FileAct:

```
<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="'SwCompression=None'"/>
                <assign to="fileDesc" from="''"/>
            </output>
            <input message="testing">
                <assign to="." from="*"/>
            </input>
        </operation>
    </sequence>
</process>
```

Enabling SWIFTNet Document Tracking

When you are creating or editing your SWIFTNet Client business 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.