



# webMethods SWIFTNet Module

## Installation and User's Guide

VERSION 6.0.1

webMethods, Inc.  
3930 Pender Drive  
Fairfax, VA 22030  
USA  
703.460.2500  
<http://www.webmethods.com>

webMethods Administrator, webMethods Broker, webMethods Developer, webMethods Installer, webMethods Integration Server, webMethods Mainframe, webMethods Manager, webMethods Modeler, webMethods Monitor, webMethods Workflow, webMethods Trading Networks, and the webMethods logo are trademarks of webMethods, Inc. "webMethods" is a registered trademark of webMethods, Inc.

Acrobat, Adobe, and Reader are registered trademarks of Adobe Systems Incorporated. Amdocs and ClarifyCRM are registered trademarks of Amdocs Ltd. Ariba is a registered trademark of Ariba Inc. Baan is a registered trademark of Baan Development NV. BEA is a registered trademark, and WebLogic Platform and WebLogic Server are trademarks of BEA Systems, Inc. BMC Software and PATROL are registered trademarks of BMC Software, Inc. BroadVision is a registered trademark of BroadVision, Inc. Chem eStandards and CIDX are trademarks of Chemical Industry Data Exchange. Unicenter is a registered trademark of Computer Associates International, Inc. Kenan and Arbor are registered trademarks of CSG Systems, Incorporated. SNAP-IX is a registered trademark, and Data Connection is a trademark of Data Connection Ltd. DataDirect, DataDirect Connect, and SequeLink are registered trademarks of DataDirect Technologies. D&B and D-U-N-S are registered trademarks of D&B, Inc. Hewlett-Packard, HP, HP-UX, and OpenView are trademarks of Hewlett-Packard Company. i2 is a registered trademark of i2 Technologies, Inc. AIX, AS/400, CICS, DB2, IBM, Infoprint, Informix, MQSeries, OS/390, OS/400, RACF, RS/6000, SQL/400, S/390, System/390, VTAM, and WebSphere are registered trademarks; and Communications System for Windows NT, IMS, MVS, SQL/DS, Universal Database, and z/OS are trademarks of IBM Corporation. JBoss and JBoss Group are trademarks of Marc Fleury under operation by JBoss Group, LLC. J.D. Edwards and OneWorld are registered trademarks, and WorldSoftware is a trademark of J.D. Edwards. Linux is a registered trademark of Linus Torvalds and others. X Window System is a trademark of Massachusetts Institute of Technology. Merant is a registered trademark of Merant, Inc. MetaSolv is a registered trademark of Metasolv Software, Inc. ActiveX, Microsoft, Outlook, Visual Basic, Windows, and Windows NT are registered trademarks; and SQL Server is a trademark of Microsoft Corporation. Netscape is a registered trademark of Netscape Communications Corporation. New Atlanta and ServletExec are trademarks of New Atlanta Communications, LLC. CORBA is a registered trademark of Object Management Group, Inc. UNIX is a registered trademark of Open Group. Oracle is a registered trademark of Oracle Corporation. PeopleSoft and Vantive are registered trademarks, and PeopleSoft Pure Internet Architecture is a trademark of PeopleSoft, Inc. Infranet and Portal are trademarks of Portal Software, Inc. RosettaNet is a trademark of "RosettaNet," a non-profit organization. SAP and R/3 are trademarks or registered trademarks of SAP AG. Siebel is a trademark of Siebel Systems, Inc. EJB, Enterprise JavaBeans, Java, JavaServer Pages, JDBC, JSP, J2EE, Solaris, SPARC, SPARCStation, Sun, Sun Microsystems, and SunSoft are trademarks of Sun Microsystems, Inc. SWIFT and SWIFTNet are trademarks of S.W.I.F.T. SCRL. Sybase is a registered trademark of Sybase, Inc. UCCnet is a trademark of UCCnet. VERITAS, VERITAS SOFTWARE, and VERITAS Cluster Server are trademarks of VERITAS Software. W3C is a registered trademark of World Wide Web Consortium.

All other marks are the property of their respective owners.

Copyright © 2003 by webMethods, Inc. All rights reserved, including the right of reproduction in whole or in part in any form.

Document ID: ESTD-SWIFTNET-IUG-601-20031117

# Contents

<b>About This Guide</b> .....	<b>5</b>
Document Conventions .....	5
Additional Information .....	6
<b>Chapter 1. Concepts</b> .....	<b>7</b>
What Is SWIFTNet? .....	8
What Is the webMethods SWIFTNet Module? .....	9
Client Application Functionality .....	9
Server Application Functionality .....	10
SNL Request and Response Primitives Support .....	10
Processing of SNL Primitives .....	10
webMethods SWIFTNet Module Architecture .....	11
webMethods SWIFTNet Module Packages .....	13
<b>Chapter 2. Installing the webMethods SWIFTNet Module</b> .....	<b>15</b>
Overview .....	16
System Requirements .....	16
Platform and Operating System Requirements .....	16
webMethods Software Requirements .....	16
Third-Party Software Requirements .....	16
Hardware Requirements .....	17
Installing webMethods SWIFTNet Module .....	17
Uninstalling webMethods SWIFTNet Module .....	18
<b>Chapter 3. Configuring the webMethods SWIFTNet Module</b> .....	<b>21</b>
Overview .....	22
Preparing the Server Side to Receive and Respond to Requests .....	22
Stage 1: Configure the SWIFTAlliance Gateway Server .....	22
Stage 2: Configure the SWIFTNet Module .....	23
Stage 3: Configure Your Integration Server .....	25
Preparing the Client Side to Send Requests and Receive Responses .....	27
Step 1: Install a Remote Access Client .....	27
Step 2: Add DLLs to the Integration Server CLASSPATH .....	28
Step 3: Set the Integration Server Environment Variables .....	29
Step 4: Edit the webMethods SNL Configuration File .....	29
Step 5: Invoke wm.swiftnet.client.services:swArguments .....	30

- Chapter 4. Defining Trading Networks Information ..... 31**
  - Overview ..... 32
  - Defining Trading Partner Profiles ..... 32
    - Why Are Trading Partner Profiles Important? ..... 32
    - Defining Your Enterprise Profile ..... 32
    - Defining Trading Partner Profiles ..... 33
  - Defining TN Document Types ..... 34
    - Importing the Sample TN Document Types ..... 34
    - Creating TN Document Types ..... 35
  - Defining Processing Rules ..... 37
    - Importing the Sample Processing Rules ..... 37
    - Creating Processing Rules ..... 38
  
- Chapter 5. webMethods SWIFTNet Module Services ..... 41**
  - WmSWIFTNetClient Package ..... 42
  - WmSWIFTNetClientSample Package ..... 49
  - WmSWIFTNetServer Package ..... 50
  - WmSWIFTNetServerSample Package ..... 54
  
- Appendix A. Configuring Transport Protocols ..... 55**
  - Overview ..... 56
  - Using MQSeries to Communicate with SWIFT ..... 56
    - Configuring the webMethods MQSeries Adapter ..... 56
  - Using File Transfer Agent to Communicate with SWIFT ..... 58
    - Configuring FTA for Inbound Messages ..... 58
    - Configuring FTA for Outbound Messages ..... 59
  
- Index..... 61**

## About This Guide

This guide describes how to install, configure, and use the webMethods SWIFTNet Module, which supports synchronous communication of SWIFT messages and files between client and server applications.

To use this guide effectively, you should:

- Have a basic knowledge of SWIFT and SWIFT terminology. For more information, go to <http://www.swift.com>.
- Have installed all necessary SWIFT software. You must work with SWIFT to determine the appropriate software needs for your company.
- Have installed the webMethods Integration Server, the webMethods Trading Networks (server side and console side) software, and the webMethods SWIFTNet Module software. For more information about installing the non-SWIFT components, see the *webMethods Integration Platform Installation Guide*.
- Be familiar with the webMethods Integration Server, the Server Administrator, and webMethods Developer and understand the concepts and procedures described in the *webMethods Integration Server Administrator's Guide* and the *webMethods Developer User's Guide*.
- Be familiar with Trading Networks Console and understand the concepts and procedures described in the various *webMethods Trading Networks* guides.

## Document Conventions

Convention	Description
<b>Bold</b>	Identifies elements on a screen.
<i>Italic</i>	Identifies variable information that you must supply or change based on your specific situation or environment. Identifies terms the first time they are defined in text. Also identifies service input and output variables.
Narrow font	Identifies storage locations for services on the webMethods Integration Server using the convention <i>folder.subfolder.service</i> .
Typewriter font	Identifies characters and values that you must type exactly or messages that the system displays on the console.
UPPERCASE	Identifies keyboard keys. Keys that you must press simultaneously are joined with the "+" symbol.

Convention	Description
\	Directory paths use the “\” directory delimiter unless the subject is UNIX-specific.
[ ]	Optional keywords or values are enclosed in [ ]. Do not type the [ ] symbols in your own code.

## Additional Information

---

The webMethods Advantage Web site at <http://advantage.webmethods.com> provides you with important sources of information about your webMethods Integration Platform:

- **Troubleshooting Information.** The [webMethods Knowledge Base](#) provides troubleshooting information for various webMethods components.
- **Documentation Feedback.** To provide documentation feedback to webMethods, complete the [Documentation Feedback Form](#) on the [webMethods Bookshelf](#).
- **Additional Documentation.** All of the webMethods documentation is available on the [webMethods Bookshelf](#).

## Concepts

- What Is SWIFTNet? ..... 8
- What Is the webMethods SWIFTNet Module? ..... 9
- webMethods SWIFTNet Module Architecture .....11
- webMethods SWIFTNet Module Packages ..... 13

## What Is SWIFTNet?

---

SWIFTNet is SWIFT's advanced IP-based messaging solution, which provides an alternate method of transferring information to SWIFT. It consists of a portfolio of products and services enabling the secure and reliable communication of financial information and transactional data.

Messages and files exchanged between parties consist of SWIFTNetLink (SNL) *primitives*. An SNL primitive is a pair of XML documents that is passed between an application program and the SNL software on your SWIFTAlliance Gateway (SAG) server. Each primitive document pair represents the invocation of a SWIFTNet processing function and includes an 'in' document and a corresponding 'out' document. These documents are either 1) the input for a function call to SWIFTNet, or 2) the SWIFTNet output of a function call.

SWIFTNet includes three messaging services:

- **SWIFTNet InterAct.** Allows the interactive (real-time) and store-and-forward exchange of messages between parties. It provides secure communication facilities for transferring messages up to 100KB. As the foundational messaging service of SWIFTNet, the InterAct Services include not only the InterAct Request/Response primitives, but also all of the supporting SWIFTNet infrastructure.
- **SWIFTNet FileAct.** Allows the exchange of files in an automated way, supporting both interactive (real-time) and store-and-forward modes. It provides secure communication facilities for transferring files. FileAct is oriented toward transferring data larger than the InterAct payload can accommodate.
- **SWIFTNet Browse.** Enables secure browser-based access to service providers' web servers. This service provides direct access to the secure messaging features of SWIFTNet InterAct and SWIFTNet FileAct. (This set of services is not currently supported by the webMethods SWIFTNet Module.)

For more information about SWIFT and SWIFTNet, see the documentation provided by SWIFT or go to <http://www.swift.com>.



---

## What Is the webMethods SWIFTNet Module?

---

The webMethods SWIFTNet Module supports synchronous communication of SWIFT messages and files between client and server applications. A client application is one that sends a request and receives a response. A server application is one that receives a request and sends a response.



**Important!** Because client and server applications cannot co-exist in the same process, if you want to use both the client and server application services, you must install the webMethods SWIFTNet Module on two different Integration Servers. You must use only the WmSWIFTNetClient packages on one Integration Server and only the WmSWIFTNetServer packages on the other Integration Server.

The SWIFTNet Module provides client-side and server-side support for the following messaging services and capabilities:

- InterAct Services
- FileAct Services



**Important!** This release of the webMethods SWIFTNet Module does not support Store-and-Forward capabilities. Without Store-and-Forward, this module requires that both parties be online for any transfer of information. Store-and-Forward will be supported in a future release.

As mentioned earlier, InterAct and FileAct are implemented as a set of SNL primitives that are exchanged between the client or server application program and the SNL software on your SAG server. Along with its packages, the SWIFTNet Module provides two DLLs, `WmSWIFTNetClient.dll` and `WmSWIFTNetServer.dll`, that invoke the functionality of the SNL libraries to transfer the SNL primitives between the client and server applications.

For more information about the packages in this module, see “[webMethods SWIFTNet Module Packages](#)” on page 13 in this chapter.

### Client Application Functionality

The `WmSWIFTNetClient.dll` invokes functionality for a client application, which sends a request to and receives a response from a server application. When using the module with a client application, you can do the following:

- Send an InterAct request message and receive a response
- Put a file
- Get a file

## Server Application Functionality

The `WmSWIFTNetServer.dll` invokes functionality for a server application, which receives a request from and sends a response to a client application. When using the module with a server application, you can do the following:

- Receive an InterAct request message and send a response
- Accept a put file request from the client application
- Accept a get file request from the client application

For more information about the architecture of the module, see [“webMethods SWIFTNet Module Architecture” on page 11](#) in this chapter.

## SNL Request and Response Primitives Support

The webMethods SWIFTNet Module supports all of the SNL request and response primitives involved in synchronous communication between client and server applications:

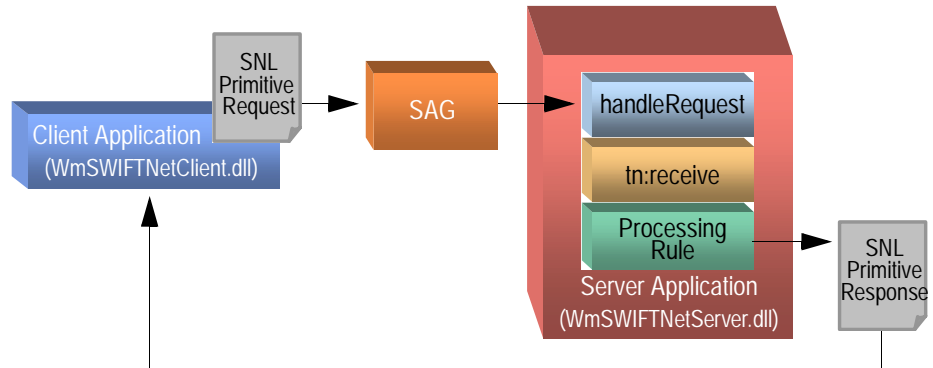
- `Sw:HandleInitRequest`
- `Sw:HandleInitResponse`
- `SwSec:CreateContextRequest`
- `SwSec:CreateContextResponse`
- `Sw:SubscribeFileEventRequest`
- `Sw:SubscribeFileEventResponse`

## Processing of SNL Primitives

When the SAG server receives a request from a client application, the `wm.swiftnet.server.services.handleRequest` service is invoked in the Integration Server on which the server application is installed. The `handleRequest` service then invokes the `wm.tn.receive` service.

Trading Networks uses the TN document types that you create to recognize the incoming request, saves the request to the Trading Networks database, and invokes one of the processing rules you created and associated with the request’s TN document type. The processing rule processes the document as necessary and generates output containing the string *xmlResponse*, which is the response to be sent back to the client application. The following diagram illustrates the process.

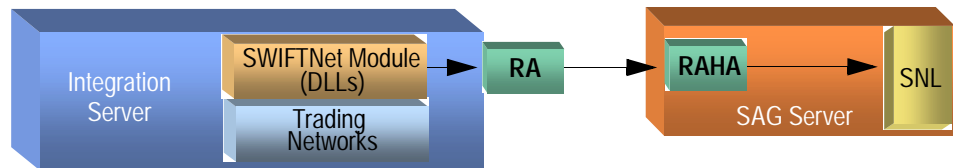
## webMethods SWIFTNet Module Processing



## webMethods SWIFTNet Module Architecture

The following diagram illustrates the components involved in transferring messages and files using the webMethods SWIFTNet Module.

## SWIFTNet Module Architecture



As indicated in the diagram, the components listed in the following table comprise and support the SWIFTNet Module.

Component	Description
Integration Server	webMethods Integration Server is the underlying server of the webMethods Integration Platform. You use the web-based user interface, Server Administrator, to manage, configure, and administer all aspects of the Integration Server, such as users, security, packages, and services.  For more information, see the <i>webMethods Integration Server Administrator's Guide</i> .

Component	Description
Trading Networks	<p>webMethods Trading Networks enables your enterprise to link with other financial institutions and marketplaces to form a business-to-business trading network. In this instance, Trading Networks enables the webMethods SWIFTNet Module to exchange messages and files with your SWIFTAlliance Gateway (SAG) server.</p> <p>For more information about Trading Networks, see the <i>webMethods Trading Networks—Getting Started with Trading Networks</i> and the <i>webMethods Trading Networks—Building Your Trading Network</i> guides.</p>
RA	<p>The Remote Access (RA) client enables the SWIFTNet Module to communicate with your SAG server and SNL through your Remote Access Host Adapter (RAHA). You must install an RA client on the same machine as the Integration Server.</p> <p>To obtain an RA client, contact SWIFT.</p>
RAHA	<p>Your RAHA enables your SAG server to exchange messages and files with the Remote Access (RA) client on your Integration Server. You must install an Remote Access Host Adapter (RAHA) on the same machine as the SAG server.</p> <p>To obtain an RAHA, contact SWIFT.</p>
SAG Server/SNL	<p>Your SWIFTAlliance Gateway (SAG) server, on which you install your SWIFTNetLink (SNL) software, must be configured to exchange messages and files with SWIFTNet. You also will use this configuration information to configure the SWIFTNet Module and your RA client.</p>



**Important!** In addition to using RAHA, you can communicate with your SAG server and SNL using MQSeries Adapter or File Transfer Agent. For more information, see [Appendix A, “Configuring Transport Protocols”](#) in this guide.

## webMethods SWIFTNet Module Packages

The webMethods SWIFTNet Module contains two sets of packages, which contain webMethods services and related files, that you install on the webMethods Integration Server. The following table describes the contents of each package.



**Important!** Because client and server applications cannot co-exist in the same process, if you want to use both the client and server application services, you must install the webMethods SWIFTNet Module on two different Integration Servers. You must use only the WmSWIFTNetClient packages on one Integration Server and only the WmSWIFTNetServer packages on the other Integration Server.

Package	Description
WmSWIFTNetClient	Contains the FileAct and InterAct services that support SWIFTNet Module client-side functionality.
WmSWIFTNetClientSample	Contains a sample implementation of the SWIFTNet Module client-side functionality.
WmSWIFTNetServer	Contains the FileAct and InterAct services that support SWIFTNet Module server-side functionality.
WmSWIFTNetServerSample	Contains a sample implementation of the SWIFTNet Module server-side functionality.

For detailed information about the contents of these packages, see [Appendix 5, “webMethods SWIFTNet Module Services”](#) in this guide.



## Installing the webMethods SWIFTNet Module

■ Overview .....	16
■ System Requirements .....	16
■ Installing webMethods SWIFTNet Module .....	17
■ Uninstalling webMethods SWIFTNet Module .....	18

## Overview



**Important!** The information in this chapter might have been updated since the guide was published. Go to the webMethods Advantage Web site at <http://advantage.webmethods.com> for the latest version of the guide.

## System Requirements

This section describes the system requirements that must be met before you can install the webMethods SWIFTNet Module.

### Platform and Operating System Requirements

The following table lists the supported platforms for the SWIFTNet Module and the Java virtual machine (JVM) to use for each platform.

Platform and Operating System	Supported JVMs
Microsoft Windows NT 4.0 SP4	JRE 1.3
Microsoft Windows 2000 and XP Professional	JRE 1.3

### webMethods Software Requirements

The following table lists the webMethods components you must install before you can install the SWIFTNet Module:

Component
webMethods Integration Server 6.0.1
webMethods Developer 6.0.1
webMethods Trading Networks 6.0.1

### Third-Party Software Requirements

Regardless of whether you are using the SWIFTNet Module for a client or server application, you must install a Remote Access (RA) client on your Integration Server. If you are using the SWIFTNet Module for a server application, you must install a Remote Access Host Adapter (RAHA) on the same machine as the SAG server. Both the RA client and the RAHA are provided by SWIFT. For more information, see your SWIFT documentation or go to <http://www.swift.com>.



## Hardware Requirements

The SWIFTNet Module does not have any requirements in excess of those for webMethods Integration Server.

## Installing webMethods SWIFTNet Module

---



**Important!** This section provides only instructions that are specific to installing the webMethods SWIFTNet Module. For complete instructions on using the webMethods Installer, see the *webMethods Integration Platform Installation Guide*.

---



**Important!** You must have administrator privileges on the webMethods Integration Server to execute these procedures. If you do not have administrator privileges, have your webMethods Integration Server administrator perform these procedures.

---

### Install the SWIFTNet Module

- 1 Download webMethods Installer 6.0.1 from the webMethods Advantage Web site at <http://advantage.webmethods.com> and start the installer.
- 2 Run the installer using the username and password you received from webMethods. Specify the installation directory as the webMethods 6.0.1 installation directory (by default, webMethods6).
- 3 In the component selection list, navigate to **webMethods Platform ▶ eStandards ▶ webMethods SWIFTNet Module** and select one of the following components to install:
  - a **SWIFTNet Module Client** to install the client component.
  - b **SWIFTNet Module Server** to install the server component.

Each component includes the following selections:

- **Documentation 6.0.1 (Optional)**. Contains the documentation for this package.
  - **Program Files 6.0.1(Required)**. Contains the program files for this package.
  - **Samples 6.0.1 (Optional, but recommended)**. Contains the services that demonstrate how to use the SWIFTNet Module services.
  - In addition, select any required webMethods components you have not installed.
- 4 Click **Next** until the installer displays the **Installation Complete** panel.

5 Click Close.



**Important!** You must configure your SWIFTNet Module and configure and start your SWIFTAlliance Gateway (SAG) server before starting the Integration Server on which you have installed the SWIFTNet Module. If at any time the SAG server goes down, you must restart the Integration Server on which you have installed the SWIFTNet Module.

---

The webMethods SWIFTNet Module starts automatically when you start the webMethods Integration Server.

## Uninstalling webMethods SWIFTNet Module

---



**Important!** This section provides only instructions that are specific to uninstalling the webMethods SWIFTNet Module. For complete instructions on using the webMethods Installer, see the *webMethods Integration Platform Installation Guide*.

---



**Important!** If you want to keep certain records or services that you use with the existing webMethods SWIFTNet Module packages on your webMethods Integration Server, make sure that you export them to a new package (from webMethods Developer, select the package to export, and select **File** ► **Export**) before performing the following procedure, which will remove all components in the packages.

---

You perform the following procedure when you want to completely uninstall the webMethods SWIFTNet Module.

### To uninstall the SWIFTNet Module from webMethods Integration Server

- 1 The instructions in the *webMethods Integration Platform Installation Guide* say to shut down all webMethods components and applications that are running on your machine. For SWIFTNet Module, you need to shut down only the Integration Server.
- 2 Select **webMethods SWIFTNet Module** as the program to uninstall.
- 3 The uninstaller removes all SWIFTNet Module-related files that were installed into the *Integration Server\_directory*\packages directory. The uninstaller does not delete files created after you installed the SWIFTNet Module (for example, user-created or configuration files), nor does it delete the directory structure that contains the files.
- 4 If you want to delete the files that the uninstaller did not delete, navigate to the *Integration Server\_directory*\packages directory and delete the related folders.

You perform the following procedure when you want to uninstall portions of the SWIFTNet Module.

### To uninstall portions of the SWIFTNet Module from webMethods Integration Server

- 1 Start the webMethods Integration Server and the Server Administrator.
- 2 Under the **Package** menu in the Server Administrator navigation area, click **Management**.
- 3 From the **Package** list, locate the following packages:
  - WmSWIFTNetClient
  - WmSWIFTNetClientSample
  - WmSWIFTNetServer
  - WmSWIFTNetServerSample
- 4 Select one of the following options for each of the desired packages:
  - Select **Delete** to delete the package without keeping a backup copy.
  - Select **Safe Delete** to remove the package and keep a backup copy. (Backup copies are stored in the *ServerDirectory*\replicate\salvage directory on the server.)
- 5 Refresh your Web browser. The selected packages are removed.



## Configuring the webMethods SWIFTNet Module

- Overview ..... 22
- Preparing the Server Side to Receive and Respond to Requests ..... 22
- Preparing the Client Side to Send Requests and Receive Responses ..... 27

## Overview

---

This chapter describes how to prepare your server application or your client application to exchange messages and files over SWIFTNet using the SWIFTNet Module.



**Important!** The following steps assume that you already have installed the webMethods Integration Server, webMethods Trading Networks, and the webMethods SWIFTNet Module, and have imported a BIC or BIC+ list. For steps to install the webMethods SWIFTNet Module, see [Chapter 2, “Installing the webMethods SWIFTNet Module”](#) in this guide. For more information about BICs, see the *webMethods SWIFT FIN Module Installation and User’s Guide*. For more information about what SWIFT software you need, work with SWIFT to determine your software needs.

---

## Preparing the Server Side to Receive and Respond to Requests

---

To prepare your server application to receive and respond to requests using the SWIFTNet Module, you must complete three stages of configuration:

- [Stage 1: Configure the SWIFTAlliance Gateway Server](#)
- [Stage 2: Configure the SWIFTNet Module](#)
- [Stage 3: Configure Your Integration Server](#)

### Stage 1: Configure the SWIFTAlliance Gateway Server

To configure your SAG server to communicate with your RA client, SWIFTNet Module, and Integration Server, complete the following steps:

- **Install a Remote Access Host Adapter (RAHA).** Install an RAHA on the same machine as the SAG server. Your RAHA enables your SAG server to exchange messages and files with the RA client on your Integration Server. To obtain an appropriate RAHA, contact SWIFT.
- **Configure your message partners**

For more information about completing these steps, see the *SWIFTAlliance Gateway File Transfer Interface Guide*, the *SWIFTAlliance Gateway Operations Guide*, and the *SWIFTAlliance Gateway Remote API Operations Guide*.



**Important!** If at any time the SAG server goes down, you must restart the Integration Server on which you have installed the SWIFTNet Module.

---

## Stage 2: Configure the SWIFTNet Module

To configure your SWIFTNet Module, complete the following steps:

- [Step 1: Install a Remote Access Client](#)
- [Step 2: Add DLLs to the Integration Server CLASSPATH](#)
- [Step 3: Set the Integration Server Environment Variables](#)
- [Step 4: Edit the webMethods SNL Configuration File](#)

### Step 1: Install a Remote Access Client

Install a Remote Access (RA) client on the same machine as the Integration Server. The RA client enables the SWIFTNet Module to communicate with your SAG server and SNL through your Remote Access Host Adapter (RAHA). To obtain an RA client, contact SWIFT.

### Step 2: Add DLLs to the Integration Server CLASSPATH

Copy the `WmSWIFTNetServer.dll` from the `webMethods6\IntegrationServer\packages\WmSWIFTNetServer` directory, and then paste the DLL into one of the directories contained in the `PATH` environment variable available to the Integration Server (preferably into the `webMethods6\IntegrationServer\lib` directory).

For the `WmSWIFTNetServer.dll` files to function properly, the DLLs listed in the following table must be copied from your SAG server and pasted into one of the folders contained in the `PATH` environment variable available to the Integration Server (preferably into the `webMethods6\IntegrationServer\lib` directory):

- |  |                                     |
|--|-------------------------------------|
| ■ <code>libftla_handler.dll</code>       | ■ <code>libsagapli_20.dll</code>    |
| ■ <code>libsagapp_20.dll</code>          | ■ <code>libsagcm_20.dll</code>      |
| ■ <code>libsagcm_util.dll</code>         | ■ <code>libsagcontrol_20.dll</code> |
| ■ <code>libsagcontrol_util_20.dll</code> | ■ <code>libsagdb.dll</code>         |
| ■ <code>libsagextractor.dll</code>       | ■ <code>libsagoh_20.dll</code>      |
| ■ <code>libsagplugin_20.dll</code>       | ■ <code>libsagprimitive.dll</code>  |
| ■ <code>libsagra_20.dll</code>           | ■ <code>libsagtypes_20.dll</code>   |
| ■ <code>libswcomm.dll</code>             | ■ <code>libswexception.dll</code>   |
| ■ <code>libswlnk.dll</code>              | ■ <code>libswpos1.dll</code>        |
| ■ <code>libswreportevt.dll</code>        | ■ <code>libswsec.dll</code>         |
| ■ <code>libswstring_20.dll</code>        | ■ <code>libswtrace.dll</code>       |

- libswtransport.dll
- libsw\_util\_xml.dll
- xerces-c\_1\_7\_0.dll

### Step 3: Set the Integration Server Environment Variables

For the `WmSWIFTNetServer.dll` file to communicate with your RA client, you must set the your system environment variables as follows:

```

ARCH=win32
GENLOG_DIR=C:\SWIFTAlliance\RA\Ra1\log
GENUTIL_DIR=C:\SWIFTAlliance\RA\bin
OSA_DIR=C:\SWIFTAlliance\RA\Ra1\log
OSA_INSTANCE=Ra1
Path=C:\SWIFTAlliance\RA\bin;C:\SWIFTAlliance\RA\lib;
PKIEXECDIR=C:\SWIFTAlliance\RA
SLP_ENV=DEFAULT
SLP_FILE=server.slp
SNL_DOMAIN_NAME=Ra1
SPK_DATA_DIR=C:\SWIFTAlliance\RA\data\pki
SWNET_BIN_PATH=C:\SWIFTAlliance\RA\Ra1\bin
SWNET_CFG_PATH=C:\SWIFTAlliance\RA\Ra1\cfg
SWNET_HOST=HOSTNAME
SWNET_HOME=C:\SWIFTAlliance\RA
SWNET_INST=Ra1
SWNET_LOG_PATH=C:\SWIFTAlliance\RA\Ra1\log
SWNET_SLP_PATH=C:\SWIFTAlliance\RA\data\
SWTRACE=C:\SWIFTAlliance\RA\Ra1\log

```

### Step 4: Edit the webMethods SNL Configuration File

At startup, the `WmSWIFTNetServer` package automatically registers itself as the server application with the SNL libraries on your SAG server and exchanges a series of pre-defined SNL primitives in sequence with your SNL libraries using your RA client. The information required to populate these primitives is retrieved from the `webMethods6\IntegrationServer\packages\WmSWIFTNetServer\config\snl.cnf` file. Following is an example of the contents of the file `snl.cnf` file:

```

# Properties for FileAct/InterAct Server
SAGMessagePartner=server
server_pki_profile=user
server_pki_password=password
Sign=FALSE
Decrypt=FALSE
Authorisation=TRUE
encryptDN=cn=DistinguishedName, o=ptsausaa, o=swift
AllFileEvents=TRUE
FullFileStatus=TRUE
SwEventEP=File_Status_Event_EP
cryptoMode=Automatic

```



For the SWIFTNet Module to exchange information with your SAG server, you must set the parameters in this file using the information you used to configure your SAG server in “[Stage 1: Configure the SWIFTAlliance Gateway Server](#)” on page 22 in this chapter and as indicated in the following table.

Set this parameter...	To...
<i>SAGMessagePartner</i>	The “Server” message partner defined in your SAG server.
<i>server_pki_profile</i>	The user name of the “Server” profile defined in your SAG server.
<i>server_pki_password</i>	The password associated with the user name of the “Server” profile defined in your SAG server. This is used to unlock the security information in your SAG server.
<i>Sign, Decrypt, and Authorisation</i>	The value is used to populate the SwSec:CreateContextRequest primitive exchanged during server initialization. Valid values: <code>True</code> and <code>False</code> .
<i>encryptDN</i>	The Distinguished Name to be used for encryption operations.
<i>AllFileEvents</i> and <i>FullFileEvents</i>	The value is used to populate the Sw:SubscribeFileEventRequest primitive exchanged during server initialization. Valid values: <code>True</code> and <code>False</code> .
<i>SwEventEP</i>	The end point defined in your SAG server to handle FileAct operations. This value is used to populate the Sw:SubscribeFileEventRequest primitive exchanged during server initialization.
<i>cryptoMode</i>	The value specifies whether your SAG server automatically performs crypto operations. Valid values: <code>Automatic</code> and <code>Manual</code> .

### Stage 3: Configure Your Integration Server

To configure your Integration Server, start the Integration Server on which you have installed the SWIFTNet Module and complete the following steps:

- [Step 1: Define Trading Partner Profiles](#)
- [Step 2: Define TN Document Types](#)
- [Step 3: Create Mapping Services](#)
- [Step 4: Define Processing Rules](#)

### Step 1: Define Trading Partner Profiles

In the webMethods Trading Networks Console, you define the trading partner profiles for yourself and all financial institutions with whom you want to exchange messages and files.

For more information about defining trading partner profiles for use with the SWIFTNet Module, see [Chapter 4, “Defining Trading Networks Information”](#) in this guide.

### Step 2: Define TN Document Types

TN document types are definitions that tell webMethods Trading Networks how to identify the incoming SNL request primitives and specify the processing rule you want to use to process the document. You must create a TN document type for each type of request you will be handling.

The SWIFTNet Module provides sample TN document types in the `webMethods6\IntegrationServer\packages\WmSWIFTNetServerSample\config\ServerDocTypes.xml` file. You can import these TN document types into Trading Networks and then modify them or use them to create new TN document types.

For information about importing and creating TN document types, see [Chapter 4, “Defining Trading Networks Information”](#) in this guide. For general information about using TN document types, see the *webMethods Trading Networks--Building Your Trading Network* guide.

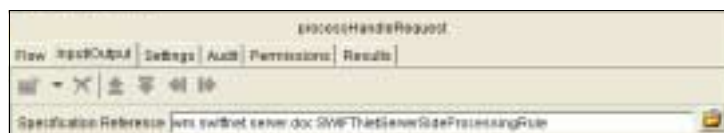
### Step 3: Create Mapping Services

To perform InterAct and FileAct operations, SNL primitives must be invoked in particular sequences. A mapping service defines in what order to invoke the SNL primitives, how to process the request, and what response to send back to the client application. You must create a mapping service for each type of request you will be handling.

The webMethods SWIFTNet Module provides sample services in the `webMethods6\IntegrationServer\packages\WmSWIFTNetServerSample` directory that you can use or on which you can base new services.

To function properly, all of the mapping services you create must meet the following requirements:

- The input/output signature must conform to the specification `wm.swiftnet.server.doc.SWIFTNetServerSideProcessingRule` in the `WmSWIFTNetServer` package. To do so, in Developer specify the `SWIFTNetServerSideProcessingRule` service in the **Specification Reference** field on the **Input/Output** tab of the mapping service as illustrated in the following figure.



- The output must contain a string *xmlResponse*. This is the response to the incoming request that the server application sends back to the client application.

For more information about this service, see [Appendix 5, “webMethods SWIFTNet Module Services”](#) in this guide.

#### Step 4: Define Processing Rules

Processing rules enable you to process incoming requests. You assign a processing rule to a particular TN document type so that when Trading Networks identifies a request, it processes that request according to the settings in the processing rule, including which mapping service should be invoked. You must create a processing rule for each type of request you will be handling.

The SWIFTNet Module provides sample processing rules in the `webMethods6\IntegrationServer\packages\WmSWIFTNetServerSample\config\ServerProcessingRules.xml` file. You can import these processing rules into Trading Networks and then modify them or use them to create new processing rules.

For information about importing and creating processing rules, see [Chapter 4, “Defining Trading Networks Information”](#) in this guide. For general information about using processing rules, see the *webMethods Trading Networks--Building Your Trading Network* guide.

You now are ready to receive requests and send responses.

## Preparing the Client Side to Send Requests and Receive Responses

---

To prepare your client application to exchange messages and files over SWIFTNet using the webMethods SWIFTNet Module, you must complete the following steps:

- [Step 1: Install a Remote Access Client](#)
- [Step 2: Add DLLs to the Integration Server CLASSPATH](#)
- [Step 3: Set the Integration Server Environment Variables](#)
- [Step 4: Edit the webMethods SNL Configuration File](#)
- [Step 5: Invoke `wm.swiftnet.client.services:swArguments`](#)

### Step 1: Install a Remote Access Client

Install a Remote Access (RA) client on the same machine as the Integration Server. The RA client enables the SWIFTNet Module to communicate with the SAG server and SNL through the Remote Access Host Adapter (RAHA). To obtain an RA client, contact SWIFT.

## Step 2: Add DLLs to the Integration Server CLASSPATH

Copy the `WmSWIFTNetClient.dll` from the `webMethods6\IntegrationServer\packages\WmSWIFTNetServer` directory, and then paste the DLL into one of the directories contained in the PATH environment variable available to the Integration Server (preferably into the `webMethods6\IntegrationServer\lib` directory).

For the `WmSWIFTNetClient.dll` files to function properly, the DLLs listed in the following table must be copied from your SAG server and pasted into one of the folders contained in the PATH environment variable available to the Integration Server (preferably into the `webMethods6\IntegrationServer\lib` directory):

- |  |                                     |
|--|-------------------------------------|
| ■ <code>libftla_handler.dll</code>       | ■ <code>libsagapli_20.dll</code>    |
| ■ <code>libsagapp_20.dll</code>          | ■ <code>libsagcm_20.dll</code>      |
| ■ <code>libsagcm_util.dll</code>         | ■ <code>libsagcontrol_20.dll</code> |
| ■ <code>libsagcontrol_util_20.dll</code> | ■ <code>libsagdb.dll</code>         |
| ■ <code>libsagextractor.dll</code>       | ■ <code>libsagoh_20.dll</code>      |
| ■ <code>libsagplugin_20.dll</code>       | ■ <code>libsagprimitive.dll</code>  |
| ■ <code>libsagra_20.dll</code>           | ■ <code>libsagtypes_20.dll</code>   |
| ■ <code>libswcomm.dll</code>             | ■ <code>libswexception.dll</code>   |
| ■ <code>libswlnk.dll</code>              | ■ <code>libswpos1.dll</code>        |
| ■ <code>libswreportevt.dll</code>        | ■ <code>libswsec.dll</code>         |
| ■ <code>libswstring_20.dll</code>        | ■ <code>libswtrace.dll</code>       |
| ■ <code>libswtransport.dll</code>        | ■ <code>libsw_util_xml.dll</code>   |
| ■ <code>xerces-c_1_7_0.dll</code>        |                                     |

### Step 3: Set the Integration Server Environment Variables

For the `WmSWIFTNetClient.dll` file to communicate with your RA client, you must set the your system environment variables as follows:

```

ARCH=win32
GENLOG_DIR=C:\SWIFTAlliance\RA\Ra1\log
GENUTIL_DIR=C:\SWIFTAlliance\RA\bin
OSA_DIR=C:\SWIFTAlliance\RA\Ra1\log
OSA_INSTANCE=Ra1
Path=C:\SWIFTAlliance\RA\bin;C:\SWIFTAlliance\RA\lib;
PKIEXECDIR=C:\SWIFTAlliance\RA
SLP_ENV=DEFAULT
SLP_FILE=server.slp
SNL_DOMAIN_NAME=Ra1
SPK_DATA_DIR=C:\SWIFTAlliance\RA\data\pki
SWNET_BIN_PATH=C:\SWIFTAlliance\RA\Ra1\bin
SWNET_CFG_PATH=C:\SWIFTAlliance\RA\Ra1\cfg
SWNET_HOST=HOSTNAME
SWNET_HOME=C:\SWIFTAlliance\RA
SWNET_INST=Ra1
SWNET_LOG_PATH=C:\SWIFTAlliance\RA\Ra1\log
SWNET_SLP_PATH=C:\SWIFTAlliance\RA\data\
SWTRACE=C:\SWIFTAlliance\RA\Ra1\log
    
```

### Step 4: Edit the webMethods SNL Configuration File

The information required to send SNL primitives to the server application is retrieved from the configuration `webMethods6\IntegrationServer\packages\WmSWIFTNetClient\config\snl.cnf` file. Following is an example of the contents of the file `snl.cnf` file:

```

# Properties for FileAct/InterAct Server
SAGMessagePartner=client
server_pki_profile=user
server_pki_password=password
encryptDN=cn=DistinguishedName, o=ptsausaa, o=swift
cryptoMode=Automatic
    
```

For the SWIFTNet Module to exchange information with the SAG server, you must set the parameters in this file as indicated in the following table.

Set this parameter...	To...
<i>SAGMessagePartner</i>	A “Client” message partner defined in the SAG server.
<i>server_pki_profile</i>	The user name of a “Client” profile defined in the SAG server.
<i>server_pki_password</i>	The password associated with the user name of a “Client” profile defined in the SAG server. This is used to unlock the security information in the SAG server.

Set this parameter...	To...
<i>encryptDN</i>	The Distinguished Name to be used for encryption operations.
<i>cryptoMode</i>	The value specifies whether the SAG server automatically performs crypto operations. Valid values: <code>Automatic</code> and <code>Manual</code> .

## Step 5: Invoke `wm.swiftnet.client.services:swArguments`

Before exchanging any primitives with the SAG server using the SWIFTNet Module, the client application must invoke the `wm.swiftnet.client.services:swArguments` service in the `WmSWIFTNetClient` package at least once. The only parameter required to be passed to this service is *SAGMessagePartner*, which is the message partner defined as the “Client” in your SAG server during configuration. The format in which this argument has to be passed is illustrated in the `wm.swiftnet.client.sample.fileAct:swExchangeFile` service in the `WmSWIFTNetClientSample` package.

For more information about these services, see [Appendix 5, “webMethods SWIFTNet Module Services”](#) in this guide.

## Defining Trading Networks Information

- Overview ..... 32
- Defining Trading Partner Profiles ..... 32
- Defining TN Document Types ..... 34
- Defining Processing Rules ..... 37

## Overview

---

Trading partner profiles help define how you and your trading partners exchange SWIFT messages and files. TN document types enable webMethods Trading Networks to identify a type of business document and specify what to extract from the business document. Processing rules tell Integration Server how to process the incoming requests for server applications.

This chapter provides you with information about defining trading partner profiles, TN document types, and processing rules in webMethods Trading Networks.

## Defining Trading Partner Profiles

---

A trading partner is any person or organization with whom you want to conduct business electronically. In the webMethods SWIFTNet Module, a trading partner is defined by several criteria that you specify in a trading partner profile, including company name and identifying information, contact information, and preferred delivery methods.

In addition to specifying trading partner profiles for all of your trading partners, you must specify a profile for your own organization.

### Why Are Trading Partner Profiles Important?

Your trading partner profiles define how you exchange messages and files with your partners. In fact, the concise definition of profiles and the configuration of processing rules are what enable you to interact successfully with your trading partners.

For the SWIFTNet Module, you define a single trading partner profile for yourself (**My Enterprise**). You also must define a trading partner profile for each trading partner with whom you will be exchanging messages and files.

### Defining Your Enterprise Profile

Before defining your trading partner profiles in Trading Networks and exchanging messages and files, you first must define your enterprise (**My Enterprise**) profile by completing the fields in the Profile Assistant in the Trading Networks Console.

For procedural information about defining your enterprise profile as well as descriptions of all the fields you must complete when defining your enterprise profile, see the *webMethods Trading Networks--Building Your Trading Network* guide. The following section provides you with the fields that you are required to complete when defining your enterprise profile for use with the SWIFTNet Module.



## Required Profile Fields

Profile information is displayed on the Trading Networks Console Profile Assistant **Corporate** tab. The following table lists and describes the required fields you must complete when defining your enterprise profile for use with the SWIFTNet Module.

Required Profile Field for My Enterprise	Description
Corporation Name	The name of your enterprise.
External ID Type	BIC
External ID Type Value	Your enterprise's BIC.

For descriptions of other fields you must complete when you defining your enterprise profile, see the *webMethods Trading Networks--Building Your Trading Network* guide.

## Defining Trading Partner Profiles

Each trading partner with whom you want to exchange messages and files must have a trading partner profile in Trading Networks. After you have defined your enterprise profile, you are ready to define your trading partners' profiles.

You define a trading partner profile by completing the fields in the Profile Assistant in the Trading Networks Console.

For procedural information about defining a trading partner profile as well as descriptions of the fields you must complete when defining a trading partner profile, see the *webMethods Trading Networks--Building Your Trading Network* guide. The following section provides you with the fields that you are required to complete when defining your trading partner profiles for use with the SWIFTNet Module.

## Required Profile Fields

Profile information displays on the Trading Networks Console Profile Assistant **Corporate** tab. The following table lists and describes the required fields you must complete when defining a trading partner profile.

Required Profile Field for Trading Partner	Description
Corporation Name	The name of the trading partner.
External ID Type	BIC
External ID Type Value	Your trading partner's BIC.

For descriptions of other fields you must complete when you define a trading partner profile, see the *webMethods Trading Networks--Building Your Trading Network* guide.

## Defining TN Document Types



TN document types are definitions that tell webMethods Trading Networks how to identify the incoming SNL request primitives and specify the processing rule you want to use to process the document. You must create a TN document type for each type of request you need to handle.

When the webMethods SWIFTNet Module receives a request, it invokes a Trading Networks service to recognize the incoming requests (SWIFTNet primitives) for server applications using the TN document types that you created. When Trading Networks recognizes the TN document type of the incoming request, Trading Networks extracts specific pieces of information from the business document based on the attributes specified in the TN document type.

The SWIFTNet Module provides sample TN document types in the `webMethods6\IntegrationServer\packages\WmSWIFTNetServerSample\config\ServerDocTypes.xml` file. You can import these TN document types into Trading Networks and then modify them or create new TN document types. For more information about TN document types, see the *webMethods Trading Networks--Building Your Trading Network* guide.

### Importing the Sample TN Document Types

#### To import the sample TN document types

- 1 Start the Integration Server, Server Administrator, and Trading Networks Console, if they are not already running.
- 2 In the Trading Networks Console, click **File ▶ Import**.
- 3 To select the TN document type file, click .
- 4 Navigate to `webMethods6\IntegrationServer\packages\WmSWIFTNetServerSample\config\ServerDocTypes.xml`, and click **Open**. The TN document types are listed on the **Import Data** screen.
- 5 Click  to select all of the TN document types and then click **OK**. The TN document types are imported.
- 6 Now you can copy and modify these TN document types as necessary.

For more information about TN document types, see the *webMethods Trading Networks--Building Your Trading Network* guide.

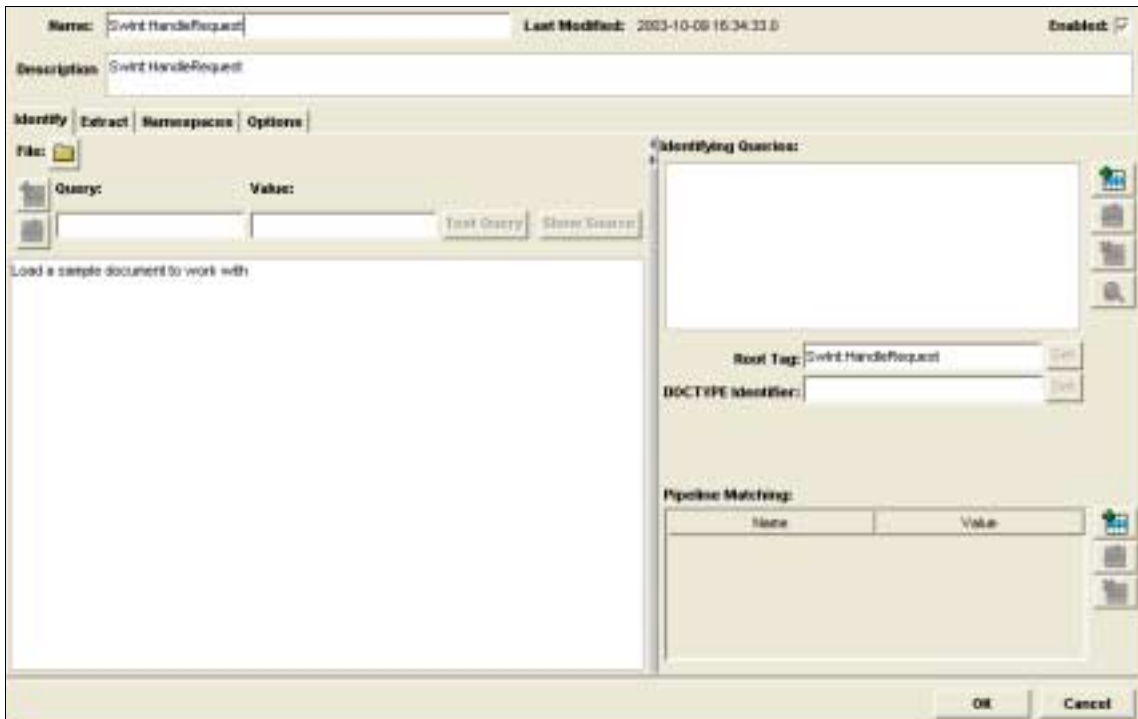
## Creating TN Document Types

When you define an internal TN document type, you specify the root tag from within the SNL primitive that the TN document type is to match and define the four prefixes that the TN document type should recognize.


### To create an internal TN document type

- 1 Start the Integration Server, Server Administrator, and Trading Networks Console, if they are not already running.
- 2 In the Selector Panel, click **My Enterprise**.
- 3 Select **View ▶ Document Types**.
- 4 Select **Types ▶ New**.
- 5 In the **Create New DocType** dialog box, select the **XML** document type category from the list, and click **OK**.
- 6 In the **Document Type Details** screen, in the **Name** field, type the name you want to give to the internal TN document type.
- 7 In the **Description** field, type a description for the internal TN document type.
- 8 On the **Identify** tab, in the **Root Tag** field, type the value of the root tag of your internal document. For example, `SwInt:HandleRequest`.

The following figure illustrates the **Identify** tab of the **Document Type Details** screen with the necessary fields completed.

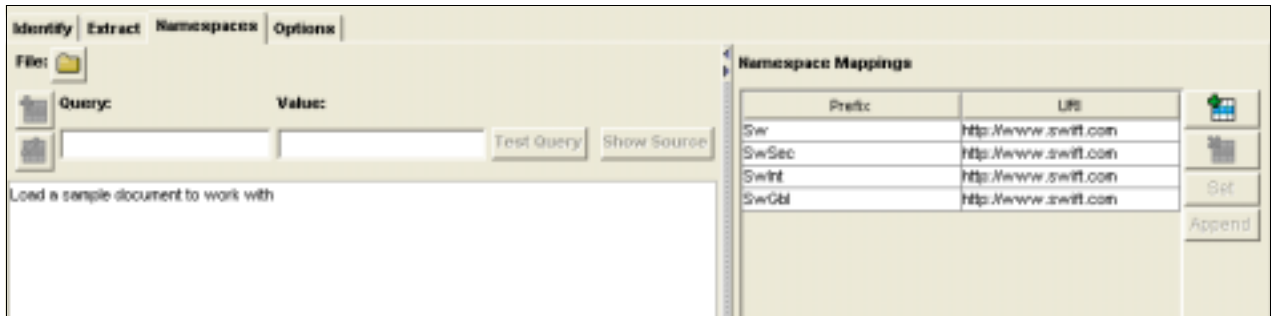


**Note:** You can tell whether a TN document type is internal or external because an external TN document type *always* has a pipeline matching variable of *processVersion*. The TN document type in the preceding figure has *no* pipeline matching variable, so it is an internal TN document type.

- 9 On the Namespaces tab, click  to define each of the following prefixes:
  - Sw
  - SwSec
  - SwInt
  - SwGbl

**Note:** You can enter anything in the URL field, but the prefixes must be entered exactly as listed in [step 9](#).

The following figure illustrates the Namespaces tab of the Processing Rule Details screen with the necessary fields completed.



10 Click OK.



For more information about TN document types, see the *webMethods Trading Networks--Building Your Trading Network* guide.

## Defining Processing Rules

Processing rules process the incoming requests (SNL primitives) for server applications by invoking mapping services that you create. The webMethods SWIFTNet Module provides sample processing rules in the `webMethods6\IntegrationServer\packages\WmSWIFTNetServerSample\config\ServerProcessingRules.xml` file. You can import the processing rules in this file using the Trading Networks Console and then modify them or create new processing rules.

### Importing the Sample Processing Rules

To import the processing rules

- 1 Start the Integration Server, Server Administrator, and Trading Networks Console, if they are not already running.
- 2 In Trading Networks, select **File** ► **Import**.
- 3 In the **Import** dialog box, click  to select the processing rule file.
- 4 Navigate to `webMethods6\IntegrationServer\packages\WmSWIFTNetServerSample\config\ServerProcessingRules.xml`, and click **Open**. The processing rules are listed on the **Import Data** screen.
- 5 Click  to select all of the processing rules and then click **OK**. The processing rules are imported.

## Creating Processing Rules

When you define a processing rule, you specify the mapping service to invoke to send the appropriate response back to the client application.

To create a processing rule

- 1 Start the Integration Server, Server Administrator, and Trading Networks Console, if they are not already running.
- 2 In the Selector Panel, click **My Enterprise**.
- 3 Select **View ▶ Processing Rules**.
- 4 Select **Types ▶ New**.
- 5 In the **Processing Rule Details** screen, in the **Name** field, type the name you want to give to the processing rule.
- 6 In the **Description** field, type a description for the processing rule.
- 7 On the **Criteria** tab, select the TN document type associated with the processing rule.

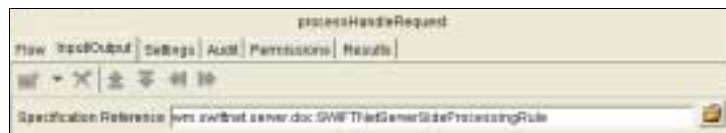
The following figure illustrates the main screen and the **Criteria** tab of the **Processing Rule Details** screen with the necessary fields completed.

The screenshot displays the 'Processing Rule Details' dialog box. At the top, the 'Name' field is filled with 'Process Swift HandleRequest', 'Last Modified' is '2003-10-06 16:32:40', and the 'Enabled' checkbox is checked. Below this is a 'Description' field containing 'Invoked when Swift HandleRequest document is received'. The 'Criteria' tab is selected, showing a section titled 'Use this rule to process documents when they match the following criteria:'. This section contains four criteria groups: 'Sender', 'Receiver', 'Document Type', and 'User Status'. Each group has radio buttons for 'Any', 'My Enterprise', and 'Unknown', and a 'Selected' list. The 'Document Type' group is expanded, showing 'Available Document Types' with 'PROFVAL\_InstRe' selected and 'Selected Document Types' with 'Swift HandleRequest' selected. At the bottom, there is a 'Recognition Errors' dropdown set to 'May have errors' and 'OK' and 'Cancel' buttons.

- 8 On the **Action** tab, click the **Perform the following actions** radio button.
- 9 Select the **Execute a service** check box and specify the mapping service that you want this processing rule to execute.

To function properly, the mapping service you specify must meet the following requirements:

- The input/output signature must conform to the specification `wm.swiftnet.server.doc:SWIFTNetServerSideProcessingRule` in the `WmSWIFTNetServer` package. To do so, in Developer specify the `SWIFTNetServerSideProcessingRule` service in the **Specification Reference** field on the **Input/Output** tab of the mapping service as illustrated in the following figure:



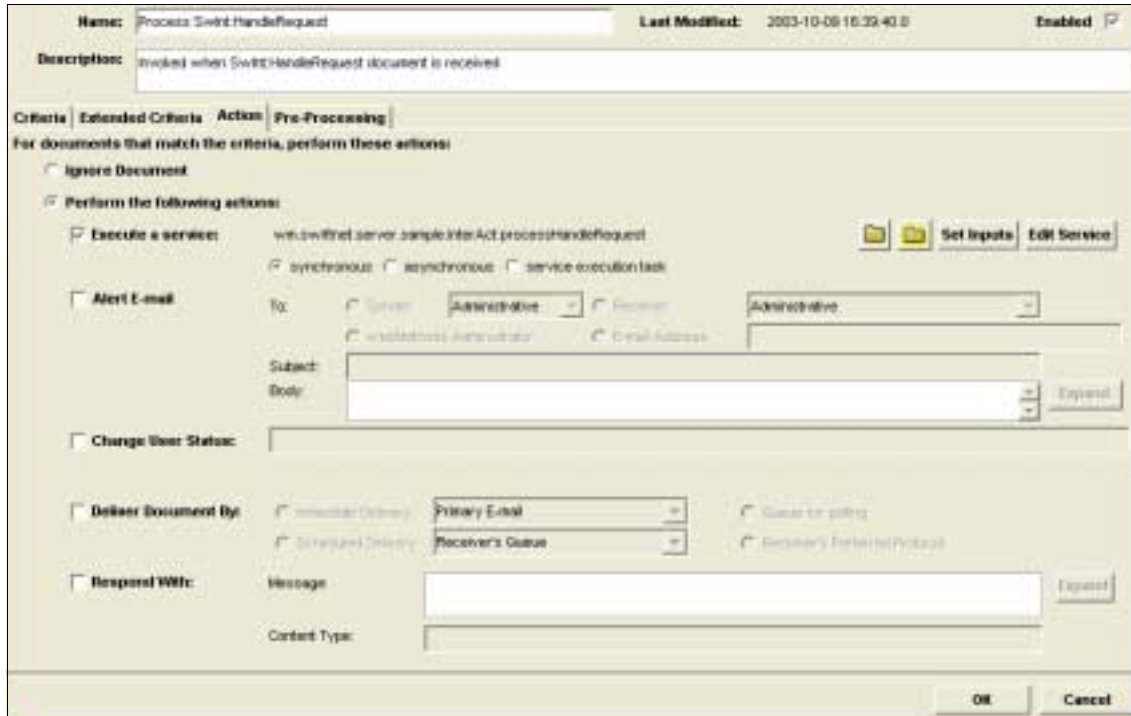
For more information about this service, see [Appendix 5, “webMethods SWIFTNet Module Services”](#) in this guide.

- The output must contain a string `xmlResponse`. This is the response to the incoming request that the server application sends back to the client application.

The SWIFTNet Module provides sample services in the `webMethods6\IntegrationServer\packages\WmSWIFTNetServerSample` directory that you can use or on which you can base new services.

10 Click OK.

The following figure illustrates the main screen and the Action tab of the Processing Rule Details screen with the necessary fields completed.





## webMethods SWIFTNet Module Services

■ WmSWIFTNetClient Package .....	42
■ WmSWIFTNetClientSample Package .....	49
■ WmSWIFTNetServer Package .....	50
■ WmSWIFTNetServerSample Package .....	54



**Important!** Because client and server applications cannot co-exist in the same process, if you want to use both the client and server application services, you must install the webMethods SWIFTNet Module on two different Integration Servers. You must use only the WmSWIFTNetClient packages on one Integration Server and only the WmSWIFTNetServer packages on the other Integration Server.

## WmSWIFTNetClient Package

---

This package contains the FileAct and InterAct services that support webMethods SWIFTNet Module client-side functionality.

### wm.swiftnet.client.doc

This folder contains the NS records representing the SNL primitives exchanged for FileAct and InterAct operations.

### wm.swiftnet.client.init

This folder contains services that perform initialization routines upon startup.

#### wm.swiftnet.client.init:startup

This service loads `WmSWIFTNetClient.dll` into memory. This DLL must be in one of the folders present in "PATH" system environment variable. Ideally it should be present in the `webMethods6\IntegrationServer\lib` folder.

### wm.swiftnet.client.services

This folder contains services that exchange SNL primitives with SAG over RA. The service `wm.swiftnet.client.services.swArguments` must have been invoked prior to invoking any other services in this folder. The services in this folder can be invoked in a predefined sequence to perform a FileAct or an InterAct operation such as putting a file, sending an interactive message, etc. In essence the services in this folder are the building blocks to perform higher level FileAct and InterAct operations.

#### wm.swiftnet.client.services:createContextRequest

This service sends the `SwSec:CreateContextRequest` to SAG over RA and returns the `SwSec:CreateContextResponse` received from SAG. This service requests SAG to create a security context.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwSecCreateContextRequest</i>	IData	Request to create a security context.
<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwSecCreateContextResponse</i>	IData	Status indicating success or failure of security context creation in SAG.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	Optional - Error details received from SAG.

**wm.swiftnet.client.services:destroyContextRequest**

This service sends the SwSec:DestroyContextRequest to SAG over RA and returns the SwSec:DestroyContextResponse received from SAG. This service requests SAG to destroy a security context.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwSecDestroyContextRequest</i>	IData	Request to destroy a security context.
<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwSecDestroyContextResponse</i>	IData	Status indicating success or failure of security context creation in SAG.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	<b>Optional</b> - Error details received from SAG.

**wm.swiftnet.client.services:exchangeFileRequest**

This service sends the Sw:ExchangeFileRequest to SAG over RA and returns the Sw:ExchangeFileResponse received from SAG. This service requests SAG to perform a FileAct operation (a get file or a put file). The information whether to put a file or get a file is specified in the Sw:ExchangeFileRequest primitive.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwExchangeFileRequest</i>	IData	Request to perform a FileAct operation.
<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwExchangeFileResponse</i>	IData	Status indicating success or failure of the FileAct operation.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	<b>Optional</b> - Error details received from SAG.

**wm.swiftnet.client.services:exchangeRequest**

This service sends the Sw:ExchangeRequest to SAG over RA and returns the Sw:ExchangeResponse received from SAG. This service requests SAG to send an InterAct message.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwExchangeRequest</i>	IData	Request to exchange a synchronous request.

<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwExchangeResponse</i>	IData	Synchronous response received.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	<b>Optional</b> - Error details received from SAG.

**wm.swiftnet.client.services:getFileStatusRequest**

This service sends the Sw:GetFileStatusRequest to SAG over RA and returns the Sw:GetFileStatusResponse received from SAG.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwGetFileStatusRequest</i>	IData	Request to get the status of a file transfer.

<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwGetFileStatusResponse</i>	IData	File transfer status response.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	<b>Optional</b> - Error details received from SAG.

**wm.swiftnet.client.services:initRequest**

This service sends the Sw:InitRequest to SAG over RA and returns the Sw:InitResponse received from SAG. This is the initialization primitive exchanged before any other primitives are exchanged.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwInitRequest</i>	IData	Initialization request primitive.

Output Variables	Type	Description
<i>SwInitResponse</i>	IData	Initialization primitive response.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	<b>Optional</b> - Error details received from SAG.

#### wm.swiftnet.client.services:sendRequest

This service sends the SwInt:SendRequest to SAG over RA and returns the SwInt:SendResponse received from SAG. This is the asynchronous version of Sw:ExchangeRequest primitive.

Input Variables	Type	Description
<i>SwIntSendRequest</i>	IData	Asynchronous request primitive.

Output Variables	Type	Description
<i>SwIntSendResponse</i>	IData	Response received from the RA client, not SAG.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	<b>Optional</b> - Error details received from the RA client.

#### wm.swiftnet.client.services:sendSynchronousRequest

This service formats the input request primitive into an XML string and then invokes the `wm.swiftnet.client.services:swCall` service to send the request primitive to SAG over RA. The response XML string received is then formatted into the appropriate response primitive.

Input Variables	Type	Description
<i>requestDocument</i>	IData	Request primitive to be sent.
<i>requestDocNSName</i>	IData	NS record name of request primitive.
<i>responseDocNSName</i>	IData	NS record name of response primitive.

Output Variables	Type	Description
<i>responseDocument</i>	IData	Response primitive received.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	<b>Optional</b> - Error details received from SAG.

**wm.swiftnet.client.services:signEncryptRequest**

This service sends the SwSec:SignEncryptRequest to SAG over RA and returns the SwSec:SignEncryptResponse received from SAG.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwSecSignEncryptRequest</i>	IData	Request sign and/or encrypt payload.

<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwSecSignEncryptResponse</i>	IData	Response received from SAG.
<i>error</i>	String	Whether an error occurred. Valid values: true and false.
<i>errorXMLString</i>	String	<b>Optional</b> - Error details received from SAG.

**wm.swiftnet.client.services:swArguments**

This service initializes the client application with SNL libraries. The service a String[] of arguments as input. The only mandatory parameter to be passed is the *SAGMessagePartner* defined in SAG. This service invokes the “SwArguments()” function defined in the SNL libraries.

An example usage would be:

```
args[0] = "WmSWIFTNetClient"  
args[1] = "-SagMessagePartner"  
args[2] = "<message partner name defined in SAG>"
```

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>args</i>	String	Initialization arguments to be passed to the SNL libraries.

**wm.swiftnet.client.services:swCall**

This service invokes the “SwCall()” function in the SNL libraries to send a request primitive to SAG. The response primitive received from SAG is then output to the pipeline.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>xmlRequest</i>	String	Request primitive to be sent to SAG.
<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>xmlResponse</i>	String	Response primitive received from SAG.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	<b>Optional</b> - Error details received from SAG.

**wm.swiftnet.client.services:termRequest**

This service sends the Sw:TermRequest to SAG over RA and returns the Sw:TermResponse received from SAG.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwTermRequest</i>	IData	Session termination request to SAG.
<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwTermResponse</i>	String	Session termination response from SAG.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	<b>Optional</b> - Error details received from SAG.

**wm.swiftnet.client.services:verifyDecryptRequest**

This service sends the SwSec:VerifyDecryptRequest to SAG over RA and returns the SwSec:VerifyDecryptResponse received from SAG.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwSecVerifyDecryptRequest</i>	IData	Request to verify a signed/encrypted message.

<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwSecVerifyDecryptResponse</i>	IData	Message decryption response from SAG.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	<b>Optional</b> - Error details received from SAG.

**wm.swiftnet.client.services:waitRequest**

This service sends the SwInt:WaitRequest to SAG over RA and returns the SwInt:WaitResponse received from SAG. This is the primitive exchanged to retrieve a response asynchronously.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwIntWaitRequest</i>	IData	Request to retrieve response asynchronously.

<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>SwIntWaitResponse</i>	IData	Asynchronous response received.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	<b>Optional</b> - Error details received from SAG.



## wm.swiftnet.client.util

This folder contains utility services.

### wm.swiftnet.client.util:formatXML

This service formats an XML string by appending the following namespaces after the root tag. The namespaces are “Sw”, “SwInt”, “SwGbl” and “SwSec”. If these namespaces are not appended to the root tag, the incoming XML response primitives cannot be converted into IData objects in the Integration Server.

Input Variables	Type	Description
<i>xmlRequest</i>	String	XML string to be formatted with namespaces.
Output Variables	Type	Description
<i>formattedXML</i>	String	XML string with namespaces appended after the root tag.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	Optional - Error details received from SAG.

## WmSWIFTNetClientSample Package

This package contains a sample implementation of the webMethods SWIFTNet Module client-side functionality.

It provides sample services for sending SNL primitives to the server application. These services call other services in the sequence required by SWIFTNet. You can use these services or use them as guides for creating your own services. This package also provides sample mapping services that map a given set of inputs into the XML primitive required by SWIFTNet.

## WmSWIFTNetServer Package

---

### wm.swiftnet.server.doc

This folder contains the NS records representing the SNL primitives exchanged for FileAct and InterAct operations.

### wm.swiftnet.server.init

This folder contains services that perform initialization routines upon startup.

#### wm.swiftnet.server.init:startup

This service loads the `WmSWIFTNetServer.dll` into memory. This DLL must be in one of the folders present in “PATH” system environment variable. Ideally it should be present in the `webMethods6\IntegrationServer\lib` folder. This service then registers Integration Server process as the server application for the message partner specified in the `webMethods6\IntegrationServer\packages\WmSWIFTNetServer\config\snl.cnf` file. The following primitives are exchanged with SAG on startup in that order.

- 1 Sw:HandleInitRequest
- 2 SwSec:CreateContextRequest
- 3 SwSec:CreateContextResponse
- 4 Sw:SubscribeFileEventRequest
- 5 Sw:SubscribeFileEventResponse
- 6 Sw:HandleInitResponse

In addition the “SwCallBack()” function in `WmSWIFTNetServer.dll` is registered as the call back function with SNL libraries.

## wm.swiftnet.server.property

This folder contains services that load properties specified in the `webMethods6\IntegrationServer\packages\WmSWIFTNetServer\config\snl.cnf` file.

### wm.swiftnet.server.property:getCommonProperties

This service retrieves the most commonly used properties from the `webMethods6\IntegrationServer\packages\WmSWIFTNetServer\config\snl.cnf` file.

Output Variables	Type	Description
<i>SAGMessagePartner</i>	String	Must correspond to a “Server” type message partner defined in SAG.
<i>server_pki_profile</i>	String	User name of the profile defined in SAG.
<i>server_pki_password</i>	String	Password associated with the user name used to unlock the security information in SAG.
<i>Sign,Decrypt and Authorization</i>	String	Values are used for populating SwSec:CreateContextRequest primitive exchanged during server initialization. Valid values: <code>True</code> and <code>False</code> .
<i>encryptDN</i>	String	Distinguished Name to be used for encryption operations.
<i>AllFileEvents, FullFileEvents</i>	String	Values are used for populating Sw:SubscribeFileEventRequest primitive exchanged during server initialization. Valid values: <code>True</code> and <code>False</code> .
<i>SwEventEP</i>	String	End point defined in SAG to handle FileAct operations. This value is used for populating Sw:SubscribeFileEventRequest primitive exchanged during server initialization.
<i>cryptoMode</i>	String	Specifies whether crypto operations are performed automatically by SAG/SNL. Valid values: <code>Automatic</code> and <code>Manual</code> .

**wm.swiftnet.server.property:getProperty**

This service retrieves the value of the specified property from the `webMethods6\IntegrationServer\packages\WmSWIFTNetServer\config\sn1.cnf` file.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>propertyName</i>	String	Property value to be retrieved.

<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>value</i>	String	Value of the property.

**wm.swiftnet.server.property:listProperties**

This service retrieves all the properties specified in `webMethods6\IntegrationServer\packages\WmSWIFTNetServer\config\sn1.cnf` file.

<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>properties</i>	IData	All properties in the <code>sn1.cnf</code> file.

**wm.swiftnet.server.property:listProperties**

This service reloads all the properties specified in the `webMethods6\IntegrationServer\packages\WmSWIFTNetServer\config\sn1.cnf` file. This could be useful if more properties are added or existing properties have been changed and the changes need to be reflected in the Integration Server immediately.

<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>properties</i>	IData	All properties in the <code>sn1.cnf</code> file.

**wm.swiftnet.server.services**

This folder contains services to handle incoming requests.

**wm.swiftnet.server.services:handleRequest**

This service is invoked by the `SwCallback` function in `WmSWIFTNetServer.dll` when a request is received from SAG. This service then recognizes the incoming request primitive as a TN document type and invokes the processing rule specified by the user. The output of the service specified by the user for the processing rule must contain the string variable “xmlResponse” which is then passed back to SAG as the response for the incoming request.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>xmlRequest</i>	String	Incoming request primitive.

Output Variables	Type	Description
<i>xmlResponse</i>	String	Outgoing response primitive.

#### wm.swiftnet.client.services:swCall

This service invokes the “SwCall()” function in the SNL libraries to send a request primitive to SAG. The response primitive received from SAG is then output to the pipeline.

Input Variables	Type	Description
<i>xmlRequest</i>	String	Request primitive to be sent to SAG.

Output Variables	Type	Description
<i>xmlResponse</i>	String	Response primitive received from SAG.
<i>error</i>	String	Whether an error occurred. Valid values: <code>true</code> and <code>false</code> .
<i>errorXMLString</i>	String	<b>Optional</b> - Error details received from SAG.

## wm.swiftnet.server.util

This folder contains utility services.

#### wm.swiftnet.server.util:formatXML

This service formats an XML string by appending the following namespaces after the root tag. The namespaces are “Sw”, “SwInt”, “SwGbl” and “SwSec”. If these namespaces are not appended to the root tag, the incoming XML response primitives cannot be converted into IData objects in the Integration Server.

Input Variables	Type	Description
<i>xmlRequest</i>	String	XML string to be formatted with namespaces.

Output Variables	Type	Description
<i>formattedXML</i>	String	XML string with namespaces appended after the root tag.

**wm.swiftnet.server.util:writeToFile**

This service writes the given string or a byte[] to the specified file name.

<u>Input Variables</u>	<u>Type</u>	<u>Description</u>
<i>FileName</i>	String	Fully qualified file name or a file name relative to the Integration Server home directory.
<i>usrdata</i>	String	<b>Optional</b> - String data to be written to the file.
<i>bytesIn</i>	byte []	<b>Optional</b> - Bytes to be written to the file.

<u>Output Variables</u>	<u>Type</u>	<u>Description</u>
<i>errorStatus</i>	String	Any error that has occurred while writing to the specified file.

## WmSWIFTNetServerSample Package

---

This package contains a sample implementation of the webMethods SWIFTNet Module server-side functionality.

It provides sample services for handling incoming SNL primitive requests that need to be processed and returned to the client. These services call other services in the sequence required by SWIFTNet. You can use these services or use them as guides for creating your own services. This package also provides sample mapping services that map a given set of inputs into the XML primitive required by SWIFTNet.

## Configuring Transport Protocols

■ Overview .....	56
■ Using MQSeries to Communicate with SWIFT .....	56
■ Using File Transfer Agent to Communicate with SWIFT .....	58

## Overview

---

The transport protocols used with the webMethods SWIFTNet Module enable you to send outbound messages to SWIFT as well as receive inbound messages from SWIFT. You can connect to SWIFT using one of the following transport protocols:

- **Remote Host Adapter.** If you are connecting to SWIFT through RAHA, see [Chapter 3, “Configuring the webMethods SWIFTNet Module”](#) in this guide.
- **IBM MQSeries.** If you are connecting to SWIFT through MQSeries Adapter, you must install the webMethods MQSeries Adapter on the Integration Server. For more information about configuring and using MQHA with the webMethods SWIFTNet Module, see [“Using MQSeries to Communicate with SWIFT” on page 56](#) in this guide.
- **FTA.** If you are using File Transfer Agent (FTA), you must configure the File Polling Listener and AFT settings in the message TPA. For more information about configuring and using the File Polling Listener with the webMethods SWIFTNet Module as well as using File Drop for outbound messages, see [“Using File Transfer Agent to Communicate with SWIFT” on page 58](#).

## Using MQSeries to Communicate with SWIFT

---

The webMethods MQSeries Adapter enables the webMethods Integration Server to send and receive messages through the MQSeries interface and systems using an IBM MQSeries message queue. This capability lets you route documents—or any piece of information—from the Integration Server to systems that use MQSeries message queuing as their information interface. The webMethods MQSeries Adapter enables you to connect to SWIFT through MQHA.



**Important!** The following procedure assumes that you already have configured your MQSeries and SAG to communicate with one another, as well as installed the webMethods MQ Adapter. For more information, see the *webMethods MQSeries Adapter User's Guide*.

---

### Configuring the webMethods MQSeries Adapter

- 1 If it is not running, start your Integration Server.
- 2 In the Server Administrator, click **WebSphereMQ...** under **Adapters**.
- 3 In the left Navigation Panel, click **Queue Manager Settings**.
- 4 Click **Add Queue Manager Settings** to define a Queue Manager using the instructions and field descriptions in the *webMethods MQSeries Adapter User's Guide*, and then click **Save**. Make sure to set the **Queue Manager Name** field to the same name that you used when defining your Queue Manager in MQSeries and SAG.



- 5 Define two queues using the following steps:
  - a Click **Queue Settings** in the left Navigation Panel.
  - b From the **Server Settings** drop-down list, select the **Queue Manager** that you defined in the previous step.
  - c Click **Find Queues on this Queue Manager**.
  - d Select the queues you defined for communication between MQSeries and SAG, and then click **Save**.
- 6 Using the instructions in the *webMethods MQSeries Adapter User's Guide*, click **Msg Handlers** in the left Navigation Panel, select your **Queue Manager**, and then click **Add Message Handler** to define two message handlers for the two queues you created in the previous step.

- a For the 'put' message handler:

Set this field...	To...
Transaction Type	IS-to-WebSphere MQ
Target Client	Non-JMS Compliant Client

- 1 Note the **Folder** and **Service Name** that you define for this handler service.
- 2 Click **Generate Service**.
- 3 Click **Continue**.
- 4 Click **Enabled**.

- b For the 'get' message handler:

Set this field...	To...
Transaction Type	WebSphere MQ-to-IS

- Under **Inbound Service Details**, click **Listen**, and then set the following fields.

Set this field...	To...
Msg Service	<code>wm.fin.transport.MQSeries:getListenerService</code>
Exception Service	The name of your exception service.
Target Client	Non-JMS Compliant Client

- Click **Generate Service**.
- c Click **Continue**.
  - d Click **Enabled**.

- In the TPA for the SWIFT message you will be sending and receiving using MQSeries, set the following fields:

Set this parameter...	To...
Transport	MQ
putMessage HandlerService	The name of the 'put' handler service, which you defined in the Service Name field.

- In Developer, navigate to the 'put' message handler service you created, and set the following:
  - `msgHeader/MsgType` variable to 8
  - `msgHeader/ReplyToQueueManager` variable to the same queue manager name that you defined in the MQSeries and SAG
  - `msgHeader/ReplyToQ` variable to the appropriate 'get' queue
  - `msgHeader/Report` variable to 3 to get reports for all of the SWIFT messages you send to SAG

## Using File Transfer Agent to Communicate with SWIFT



**Important!** To use FTA, you must have the WmFlatFile package installed, which is installed by default with the webMethods Integration Server.

File Transfer Agent (FTA) enables the webMethods Integration Server to exchange information with other systems. If you are using FTA to receive inbound SWIFT messages through the File Polling Listener, and File Drop capabilities to send outbound SWIFT messages, the File Polling Listener and the message TPA must be configured properly.

### Configuring FTA for Inbound Messages

**To configure the webMethods File Polling Listener for the webMethods SWIFTNet Module**

- In the Server Administrator, click **Security** ▶ **Ports** ▶ **Add Port**.
- Select **webMethods/FilePolling** and click **Go to Step 2**.
- Configure the File Polling Listener's general fields as described in the "Configuring Ports" section of the "Configuring the Server" chapter in the *webMethods Integration Server Administrator's Guide*.

- 4 Set the following fields so that the File Polling Listener handles SWIFT messages properly.

Set this field...	To...
Content Type	application/x-wmflatfile
Folder location	The fully qualified path of the folder in which SAG will be sending SWIFT messages. The folder must be accessible to both SAG and webMethods Integration Server.
Processing Service	wm.fin.transport.AFT:processInboundFile

## Configuring FTA for Outbound Messages

To configure FTA using File Drop capabilities to send outbound SWIFT messages using the webMethods SWIFTNet Module, you must complete the following procedure.

### To configure File Drop for the webMethods SWIFTNet Module

- 1 Map a network directory in which you want to drop files to be picked up by SAG, which can be any directory in the webMethods Integration Server.
- 2 In the TPA for the SWIFT message, set the following parameters:

Set this field...	To...
Transport	AFT
Folder location	The fully qualified path of the folder in which the webMethods Integration Server will be dropping SWIFT messages. The folder must be accessible to both SAG and the Integration Server.



# Index

## A

architecture 11

## C

client application

configuration

step 1: install a remote access client 27

step 2: add DLLs to the Integration Server CLASSPATH 28

step 3: set the Integration Server environment variables 29

step 4: edit the webMethods SNL configuration file 29

step 5: invoke `wm.swiftnet.client.services:swArguments` 30

functionality 9

configuring the client side 27

configuring the server side 22

## D

documentation

about this guide 5

additional information 6

document conventions 5

## E

enterprise profile, defining 32

required profile fields 33

## F

FTA transport protocol 58

## H

hardware requirements 17

## I

installing webMethods SWIFTNet Module 17

## M

module

architecture 11

defined 9

packages 13

WmSWIFTNetClient 13

WmSWIFTNetClientSample 13

WmSWIFTNetServer 13

WmSWIFTNetServerSamples 13

MQHA transport protocol 56

## O

operating system requirements 16

## P

packages 13

WmSWIFTNetClient 13, 42

WmSWIFTNetClientSample 13, 49

WmSWIFTNetServer 13, 50

WmSWIFTNetServerSample 54

WmSWIFTNetServerSamples 13

platform requirements 16

preparing the client side 27

preparing the server side 22

primitives, SNL

processing 10

support 10

processing rules 27

creating 37

importing samples 37

processing SNL primitives 10

## R

RAHA transport protocol 56

client side 27

server side 22

requirements

hardware 17

operating system 16

platform 16

system 16

third-party software 16

## S

server application

configuration

stage 1: configure the SWIFTAlliance Gateway Server 22

stage 2: configure the SWIFTNet Module 23

stage 3: configure your Integration Server 25

functionality 10

SNL primitives

processing 10

support 10

SWIFT Network, defined 8

SWIFTNet module

architecture 11

defined 9

packages 13

WmSWIFTNetClient 13

WmSWIFTNetClientSample 13

WmSWIFTNetServer 13

WmSWIFTNetServerSamples 13

SWIFTNet, defined 8

system requirements 16

## T

third-party software requirements 16

TN document types

creating 35

importing samples 34

trading partner profiles

defining 33

required enterprise profile fields 33

why they are important 32

transport protocols

FTA 58

MQHA 56

RAHA 56

client side 27

server side 22

## U

uninstall SWIFT FIN Module 18

## W

wm.swiftnet.client.doc folder 42

wm.swiftnet.client.init folder 42

startup service 42

wm.swiftnet.client.services folder 42

createContextRequest service 42

destroyContextRequest service 43

exchangeFileRequest service 43

exchangeRequest service 44

getFileStatusRequest service 44

initRequest service 44

sendRequest service 45

sendSynchronousRequest service 45

signEncryptRequest service 46

swArguments service 46

swCall service 47, 53

termRequest service 47

verifyDecryptRequest service 48

waitRequest service 48

wm.swiftnet.client.util folder 49

formatXML service 49

wm.swiftnet.server.doc folder 50

wm.swiftnet.server.init folder 50

startup service 50

wm.swiftnet.server.property folder 51

getCommonProperties service 51

getProperty service 52

listProperties service 52

wm.swiftnet.server.services folder 52

handleRequest service 52

wm.swiftnet.server.util folder 53

formatXML service 53

writeToFile service 54

WmSWIFTNetClient package 42

WmSWIFTNetClientSample package 49

WmSWIFTNetServer package 50

WmSWIFTNetServerSample package 54

