

Developing Connectors

© 1997–2004 edocs® Inc. All rights reserved.

edocs, Inc., One Apple Hill Dr., Natick, MA 01760

The information contained in this document is the confidential and proprietary information of edocs, Inc. and is subject to change without notice.

This material is protected by U.S. and international copyright laws. edocs and eaPost are registered in the U.S. Patent and Trademark Office.

No part of this publication may be reproduced or transmitted in any form or by any means without the prior written permission of edocs, Inc.

eaSuite, eaDirect, eaPay, eaCare, eaAssist, eaMarket, and eaXchange are trademarks of edocs, Inc.

All other trademark, company, and product names used herein are trademarks of their respective companies.

Printed in the USA.

Preface

In This Section

Using this Manual	iν
Finding the Information You Need	
If You Need Help	

Using this Manual

Welcome to Developing Connectors.

This manual helps you understand the integration architecture and how your solution can work with today's market-leading OSS software.

Before You Get Started

You should be familiar with the following:

- Your application architecture
- Programming in Java
- Designing or working with databases
- eXtended Markup Language (XML)
- Your OSS

Who Should Read this Manual

This manual is for developers and project managers who are responsible for configuring and developing connectors which allow you solution to exchange data with OSS.

However, there are other topics covered in this manual that may interest other members of the project development team.

Administrators

You will find information about running and managing connectors in this manual. You will also find detailed information about how the solution exchanges information with backend systems. This will help you not only understand the application architecture, but help you quickly pinpoint problems that may occur when communication between your solution and the OSS breaks down.

Developers

You will find everything you need to know in order to build, generate and run connectors. This manual contains instructions on how to use the Integration Logic Studio you use to build connectors quickly and easily. It also shows you how to generate the runtime files then launch the connector. If your solution requires customized data structures, you also can read about using the Message Schema Reference. This part of the connector describes the structure of messages which carry the data exchanged between your solution and the back end.

Project Architect

You can use the information in this manual to learn about connectors and how they exchange information with the backend. You will find information about the way the connector works and how you can use them to integrate data from the backend. There is also a section covering integration scenarios.

Project Manager

You will find information about connectors and the different tools used to create and run them. You will find information about how your solution uses connectors and how they exchange messages. You may also find the section about the Message Schema Reference of interest as this section deals with the structure and customization of the messages used to exchange data with the backend.

How this Manual is Organized

This manual contains the following chapters:

Overview of Integrating Account Applications

This chapter covers the fundamentals of integrating an TSM solution.

It contains information about:

- Essential concepts
- Components
- Integration strategies

Installing the SmartLink (ISF)

This chapter covers installing the SmartLink (ISF).

It contains information about installing:

- SmartLink (ISF) components
- Integration Logic Studio

Before You Start

This chapter covers what you need to know before you start building connectors.

It contains information about:

- Basic concepts of the SmartLink (ISF)
- Connector components
- Connector files
- Selecting the connector to build

Designing Your Connector

This chapter covers how to design your connector.

It contains information about:

- The basics of each connector type
- How to create a connector from one of the templates.

Building Your Connector

This chapter covers using the Integration Logic Studio (ILS).

It contains information about using the ILS to create and configure the components of a connector:

- Integration processes
- Processors
- Scripts
- Macros
- Extensions
- Parameters

Working With Connector Extensions

This chapter covers working with connector extensions in the ILS.

It contains information about:

- Creating custom components
- Using custom components

Working With the SmartLink (ISF) Message Cache

This chapter covers working with the Message Cache.

It contains information about:

- Saving messages in the cache
- Extracting messages from the cache

Working With the Message Schema Reference

This chapter covers working with the Message Schema Reference.

It contains information about:

- Adding messages
- Customizing messages
- Working with the Schema Reference tool
- Generating the Message Schema Reference files

Generating Your Connector

This chapter covers generating the runtime files of your connector.

It contains information about:

- Using the ILS to generate the runtime files
- Using the command line tool to generate runtime files

Running Your Connector

This chapter covers running the connector.

It contains information about:

- Specifying processing priorities
- Starting and stopping connectors
- Administrating connectors

SmartLink (ISF) Tool Reference

This appendix is a tool reference guide. It covers the location, configuration and use of the administration tools.

The administration tools covered include:

- Synchronizer connector tools
- OSS connector tools
- Approval Sequencer tools

Core Messages

This appendix lists available core messages.

Core Business Objects

This appendix lists the Business Objects which support additional parameters.

What Typographical Changes and Symbols Mean

This manual uses the following conventions:

TYPEFACE	MEANING	EXAMPLE
Italics	Manuals, topics or other important items	Refer to <i>Developing</i> Connectors.
Small Capitals	Software and Component names	Your application uses a database called the CID.
Fixed Width	File names, commands, paths, and on screen commands	Go to //home/my file

Preface

Obtaining edocs Software and Documentation

You can download edocs software and documentation directly from Customer Central at https://support.edocs.com. After you log in, click on the Downloads button on the left. When the next page appears, you will see a table displaying all of the available downloads. To search for specific items, select the Version and/or Category and click the Search Downloads button. If you download software, an email from edocs Technical Support will automatically be sent to you (the registered owner) with your license key information.

If you received an edocs product installation CD, load it on your system and navigate from its root directory to the folder where the software installer resides for your operating system. You can run the installer from that location, or you can copy it to your file system and run it from there. The product documentation included with your CD is in the Documentation folder located in the root directory. The license key information for the products on the CD is included with the package materials shipped with the CD.

Finding the Information You Need

The product suite comes with comprehensive documentation set that covers all aspects of building solutions based on the edocs Telco Service & Analytics Manager. You should always read the release bulletin for late-breaking information.

Getting Started

If you are new to the edocs Telco Solutions, you should start by reading *Introducing Telco Service & Analytics Manager Applications*. This manual contains an overview of the various components along with the applications and their features. It introduces various concepts and components you must be familiar with before moving on to more specific documentation. Once you have finished, you can read the manual which covers different aspects of working with the application. At the beginning of each manual, you will find an introductory chapter which covers concepts and tasks.

Designing Your Solution

While reading *Introducing Telco Service & Analytics Manager Applications*, you should think about how the different components can address your solution's needs.

You can refer to *Developing Telco Service Manager (TSM)* for information about extending the object model, application security, and other design issues. The *CID Reference Guide* also gives you the information about how the information in your solution is managed and stored.

You can refer to *Developing Telco Analytics Manager (TAM)* for information about customizing the database, synchronizing data with TSM, loading data from external invoice files, and other design issues. The *CBU Reference Guide* also gives you the information about how the information in your solution is managed and stored. You should also read the section on integrating TAM with TSM in *Developing Telco Analytics Manager (TAM)*.

You can also read the introduction of *Developing Connectors* for information about integrating your solution.

Installing Telco Service & Analytics Manager Applications

You should start by reading the Release Bulletin. For detailed installation and configuring information, refer to *Installing Telco Service & Analytics Manager Applications*. This manual covers installing applications on one or more computers. It also contains the information you need to configure the different components you install.

You might also refer to *Developing Telco Service & Analytics Manager Applications* and *Developing Connectors* as these manuals contain information on customizing applications and working with other software.

If you are upgrading, be sure to read *Migrating Telco Service & Analytics Manager Applications*.

Building Your Solution

If you are designing and programming your solution, you have several different sources of information. If you are programming the user interface of the solution, you should read *Developing User Interfaces*. You also refer to the *BLM Specification* and *JSPF specification* for detailed information about programming the user interface. For configuring the various components, you refer to *Installing Telco Service & Analytics Manager Applications* and sections in other documents which deal with the component to configure.

If you are designing and programming TAM, you have several different sources of information. If you are programming the user interface of the solution, you should read *Developing Reports*. You also refer to the *QRA API Specification* and the *QRA Configuration File Reference Documentation* for detailed information about the different components you can use to build reports. For configuring the various components, you refer to *Installing Telco Service & Analytics Manager Applications* and sections in other documents which deal with the component to configure.

If you are working with the business logic of your solution, you should read *Developing Telco Service Manager (TSM)*. You can also refer to the *BLM Reference Guide* for more information about the design and structure of the BLM object model. For information about how this information is stored, you should refer to the *CID Reference Guide* along with the *CID Reference* documentation for your database. In order to develop your application, you most likely will need to install and run the Loopback Connector. This component mimics back-end applications for development purposes. For information about installing and running this component, refer to *Using the Loopback Connector*.

If you are working on the data warehouse side of TAM, you should read *Developing Telco Analytics Manager (TAM)*. For more information about the design and structure of the CBU, you should refer to the *CBU Reference Guide* along with the *CBU Reference* documentation for your database. You should also read *Developing Telco Analytics Manager (TAM)* for information about synchronizing data between the TAM and *Telco Service Manager (TSM)*. In this manual, you will also find information about loading data in both the CBU and the CID.

For more information about integrating your application, you should read *Building Connectors* to learn how Telco Service & Analytics Manager applications work with different software.

Integrating Your Solution

If you are involved in configuring your solution to work with Operation Support Software (OSS), you should read *Building Connectors*. This manual helps you understand the integration architecture and shows you how to build connectors to connect to today's market-leading OSS software. You can also read *Using the Loopback Connector* for information about a connector built for development purposes. Other manuals you can refer to for information about configuring your application include *Introducing Telco Service & Analytics Manager Applications, Developing Telco Analytics Manager (TAM)*, and *Developing Telco Service Manager (TSM)*.

Managing Telco Service & Analytics Manager Applications

If you are responsible for managing Telco Service & Analytics Manager applications, you should read the *Installing Telco Service & Analytics Manager Applications* for information about configuring various components and information about working with different application servers. *Administrating Telco Service & Analytics Manager Applications* covers what you need to know about managing your solution at runtime. For information about OSS systems, you should read *Building Connectors*.

If You Need Help

Technical support is available to customers who have valid maintenance and support contracts with edocs. Technical support engineers can help you install, configure, and maintain your edocs application.

edocs provides global Technical Support services from the following Support Centers:

US Support Center

Natick, MA Mon-Fri 8:30am – 8:00pm US EST Telephone: 508-652-8400

Europe Support Center

London, United Kingdom Mon-Fri 9:00am – 5:00 GMT Telephone: +44 20 8956 2673

Asia Pac Rim Support Center

Melbourne, Australia Mon-Fri 9:00am – 5:00pm AU Telephone: +61 3 9909 7301

Customer Central

https://support.edocs.com

Email Support

mailto:support@edocs.com

When you report a problem, please be prepared to provide us the following information:

- What is your name and role in your organization?
- What is your company's name?
- What is your phone number and best times to call you?
- What is your e-mail address?
- In which edocs product did a problem occur?
- What is your Operating System version?
- What were you doing when the problem occurred?
- How did the system respond to the error?
- If the system generated a screen message, please send us that screen message.

If the system wrote information to a log file, please send us that log file.
 If the system crashed or hung, please tell us.

Contents

Ргетасе	111
Overview of Integrating Account Applications	22
About the SmartLink (ISF)	23
Overview of the SmartLink (ISF) Application Integration Services	24
Overview of the SmartLink (ISF) Connectors	25
Overview of the Message Schema Reference Repository Overview of the SmartLink (ISF) Tools	26 26
About Integration Architectures	20 27
Connecting to OSS Applications Systems Through an EAI Product	28
Connecting to OSS Applications Systems Through an OSS Connector	29
Connecting to OSS Applications Directly Through a MOM	30
Installing the SmartLink (ISF)	31
Installing the SmartLink (ISF)	32
Before You Start	35
About Developing Connectors	36
About the Principle Components of Connectors	37
Integration Processes	38
Macros	40
Scripts	40
Connector Program Files	41
Selecting the Template	43
Designing Your Connector	45
Overview of Designing Connectors	46
About the Synchronizer Template	47
About the OSS Connector Template	48
Overview of Modifying Existing Integration Processes	49
Overview of Creating Integration Processes	50
Building Your Connector	51
Working with Connectors	52
About Connectors	53
Working with Integration Processes	55
Working with Processors	57
About Processors	57
Working with Processors	64
Configuring Processors	66
Connecting Processors	91

Programming Scripts Working with Macros Types of Macros Specifying the Macro file Programming Macros Samples Working with Extensions Working with Parameters Working With Connector Extensions	94 95 102 102 104 105 110 113 115
About Extensions	120
Defining Custom Processors	121
Validating Custom Processors	122
Using Custom Processors	123
Generating Custom Processors	124
Samples	125
Sample Custom Outbound Queue	125
Sample Custom Inbound Queue	126
Working with the SmartLink (ISF) Message Cache	129
About the SmartLink (ISF) Message Cache	130
About the Saved SmartLink (ISF) Message Structure	131
About the SmartLink (ISF) Message Cache Script APIs	132
About the SmartLink (ISF) Message Cache Processors	133
Working With the SmartLink (ISF) Message Cache	134
Overview of Working with the SmartLink (ISF) Message Cache	134
Example of Integrating the SmartLink (ISF) Message Cache	134
Creamo me omani mk noci Wessage Cache	135
Creating the SmartLink (ISF) Message Cache Adding the SmartLink (ISF) Message Cache Processors	135 136
Adding the SmartLink (ISF) Message Cache Processors	136
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache	136 139
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages	136 139 141
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages	136 139 141 144
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages Using the SmartLink (ISF) Message Cache to Build a Business Event Queue	136 139 141 144 145
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages	136 139 141 144
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages Using the SmartLink (ISF) Message Cache to Build a Business Event Queue New Topic	136 139 141 144 145 145
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages Using the SmartLink (ISF) Message Cache to Build a Business Event Queue New Topic	136 139 141 144 145 145
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages Using the SmartLink (ISF) Message Cache to Build a Business Event Queue New Topic Creating an OSS Business Event Queue Working With the Message Schema Reference	136 139 141 144 145 145 146
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages Using the SmartLink (ISF) Message Cache to Build a Business Event Queue New Topic Creating an OSS Business Event Queue Working With the Message Schema Reference About the Message Structure Files	136 139 141 144 145 145 146 147
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages Using the SmartLink (ISF) Message Cache to Build a Business Event Queue New Topic Creating an OSS Business Event Queue Working With the Message Schema Reference About the Message Structure Files About Working with the Message Schema Reference	136 139 141 144 145 145 146 147
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages Using the SmartLink (ISF) Message Cache to Build a Business Event Queue New Topic Creating an OSS Business Event Queue Working With the Message Schema Reference About the Message Structure Files About Working with the Message Schema Reference Overview of the Business Message Structure	136 139 141 144 145 145 146 147 148 149 150
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages Using the SmartLink (ISF) Message Cache to Build a Business Event Queue New Topic Creating an OSS Business Event Queue Working With the Message Schema Reference About the Message Structure Files About Working with the Message Schema Reference Overview of the Business Message Structure About the Control Area	136 139 141 144 145 145 146 147 148 149 150 151
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages Using the SmartLink (ISF) Message Cache to Build a Business Event Queue New Topic Creating an OSS Business Event Queue Working With the Message Schema Reference About the Message Structure Files About Working with the Message Schema Reference Overview of the Business Message Structure About the Control Area About the Business Data Area	136 139 141 144 145 145 146 147 148 149 150 151 152
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages Using the SmartLink (ISF) Message Cache to Build a Business Event Queue New Topic Creating an OSS Business Event Queue Working With the Message Schema Reference About the Message Structure Files About Working with the Message Schema Reference Overview of the Business Message Structure About the Control Area About the Business Data Area Adding New Messages	136 139 141 144 145 145 146 147 148 149 150 151 152 153
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages Using the SmartLink (ISF) Message Cache to Build a Business Event Queue New Topic Creating an OSS Business Event Queue Working With the Message Schema Reference About the Message Structure Files About Working with the Message Schema Reference Overview of the Business Message Structure About the Control Area About the Business Data Area Adding New Messages To add a new message (Continued)	136 139 141 144 145 145 146 147 148 149 150 151 152 153 155
Adding the SmartLink (ISF) Message Cache Processors Connecting to the SmartLink (ISF) Message Cache Saving Messages Extracting Messages Using the SmartLink (ISF) Message Cache to Build a Business Event Queue New Topic Creating an OSS Business Event Queue Working With the Message Schema Reference About the Message Structure Files About Working with the Message Schema Reference Overview of the Business Message Structure About the Control Area About the Business Data Area Adding New Messages	136 139 141 144 145 145 146 147 148 149 150 151 152 153

Working with Synchronizer Outbound Message Fields Customizing Synchronizer Inbound Messages Customizing Standard Message Structures Generating the Message Schema Reference Files To generate message documentation files To generate message mapping definition files for Synchronizer connectors To generate message mapping definition files for OSS Connectors	169 170 171 173 173 173 174
Generating Your Connector	175
About Generating Your Connector	176
Running Your Connector	179
About Running the Connector Specifying Processing Priority Starting and Stopping the Synchronizer Connector About Synchronizer Run Modes Starting and Stopping the OSS Connector Administrating the Connector About Administrating Connectors	180 181 183 183 184 185
SmartLink (ISF) Tool Reference	189
About the SmartLink (ISF) Administration Tools Synchronizer Connector Administration Tools agentstart Syntax agentadm Syntax OSS Connector Administration Tools ossstart Syntax ossadm Syntax SmartLink (ISF) Administration Tool isfadm Syntax Message Schema Reference Tool schemarefTool Syntax Connector Generator Tool ilccGenerator Syntax Core Messages	190 191 191 192 195 195 196 196 197 199 201 201
Core Business Objects	207
About Core Business Objects About Parameters and Messages Outbound Messages Inbound Messages	208 209 209 209

Element Reference	211
Integration Processes	212
Processors	214
Message Queues	215
Inbound Batch Queue	216
Inbound Email Queue	217
Inbound File System Queue	217
Inbound JMS Queue	218
Inbound JMS Topic Queue	219
Inbound Memory Queue	220
Inbound Socket Queue	220
Outbound Email Queue	221
Outbound File System Queue	221
Outbound JMS Queue	222
Outbound JMS Topic Queue	222
Outbound Memory Queue	223
Outbound Socket Queue	223
Message Cache Polling Processor	223
Database Polling Processor	224
Request Queue	226
Request Queue Agent	227
Utilities	229
Encoding Converter	229
DOM Builder	229
DOM Serializer	229
XML Validator	230
Message Counter	230
Pause	231
Custom Processors	232
Routers	233
Script Engine	233
Message Type Filter	233
Message Broadcaster	234
MultiQueue Message Filter	234
MultiQueue Message Broadcaster	235
MultiQueue Message Tracker	235
Error Handler	236
Exception Converter	236
Retry Handler	236
CID Update Error Handler	236
BLM/XML Transmapper	237
Java/XML Transmapper	238
XML Parser	239
XSL Transmapper	240
CID Objects Builder	241

Index	247
Parameters	245
Extensions	244
Scripts	243
Macros	242

CHAPTER 1

Overview of Integrating Account Applications

In This Section

About the SmartLink (ISF)	23
About Integration Architectures	27

About the SmartLink (ISF)

The SmartLink (ISF) enables the TSM to communicate, or integrate with the core CSP technology infrastructure, for example billing platforms, Customer Relationship Management (CRM) software, and other business and operational support systems (BSS/OSS). This in turn allows the CID to maintain a synchronized cache of semi-static customer and service data, as well as passing requests and responses between the various systems.

The SmartLink (ISF) can be considered to comprise:

- Connectors that move data between TSM and the transport layer. These connectors are called Synchronizers.
- Connectors that move data between BSS/OSS applications and the transport layer.
 These connectors are called BSS/OSS Connectors.
- A repository of message definitions to build and process messages for all common transactions. This is the Message Schema Reference Repository.
- Associated tools to design and configure Connectors along with administration tools to run and manage them.

Overview of the SmartLink (ISF) Application Integration Services

When integrating TSM, you need a set of fundamental integration services to exchange information between the application and the BSS/OSS. The SmartLink (ISF) provides the application integration services needed for the TSM to communicate with any BSS/OSS system. These services include:

Message structure transformation

This involves transforming message structures from one structure to another or one format to another.

Message data mapping

This involves filling a message structure with data extracted by core or external APIs.

Message routing

This involves dispatching messages to different destinations.

Middleware interaction

This involves sending and receiving messages using middleware.

Error handling

This involves catching and managing exceptions and errors.

Call sequencing

This involves core or external API call sequencing.

Depending on the service, the SmartLink (ISF) may have either a dedicated component or a core component which helps perform the service.

Overview of the SmartLink (ISF) Connectors

The SmartLink (ISF) is not only a set of application integration services, but is a framework in which the providers of the services work together.

In order to integrate an application, these service providers, or SmartLink (ISF) components, are arranged into dataflows. This means that they are connected together and organized into a step-by-step sequence. A connector is part of the SmartLink (ISF) that is responsible for moving data between applications and transport layers.

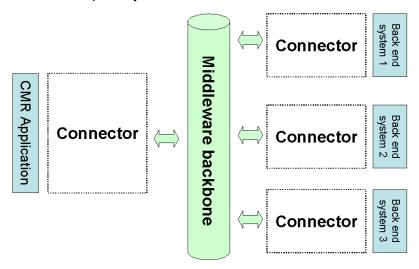
There are two types of connectors:

Synchronizers

They are responsible for data exchanged between your TSM and the transport layer

OSS Connectors

They are responsible for data exchanged between the BSS/OSS applications and the transport layer



For example, the Synchronizer connector is responsible for moving data between the TSM and the transport layer. This connector uses different connector components to carry out the following tasks:

Send data to the middleware

- 1. Extract the requests from the request queue located in the CID
- 2. Use these requests to generate XML messages
- 3. Send this XML message to the middleware backbone

Receive data from the middleware

- 4. Receive the XML message from the middleware
- 5. Use these messages to create CID objects

6. Update the CID by running scripts which use the CID objects

Overview of the Message Schema Reference Repository

The Message Schema Reference repository is a repository of message definition XML files. Each message handled by a connector has a specific definition file to manage the exchange of information.

These messages fall into one of the following categories:

REQ Messages

The connector creates these messages using the requests in the CID and sends them to the middleware.

DO Messages

The connector receives these messages from the middleware and uses them to update the CID.

System

These definition files are for other types of messages which do not fall into one of the categories above.

Each of these messages has a corresponding definition file to specify the structure of the message and its mapping definitions.

Overview of the SmartLink (ISF) Tools

The SmartLink (ISF) comes with a set of tools to help you create and manage your connectors. The tools include:

The Integration Logic Studio

This tool is for designing connectors. Its easy-to-use interface makes it easy to design and configure your connector.

The Connector Administration tools

These tools are for running and managing the connectors.

About Integration Architectures

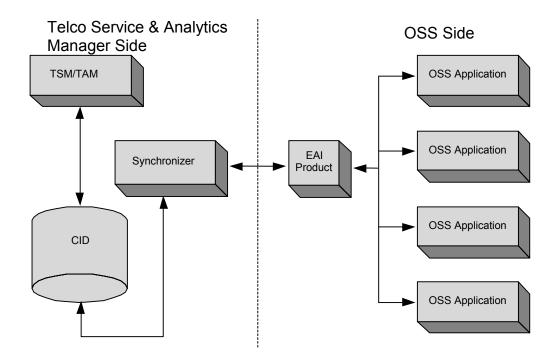
Because of the many different technological infrastructures deployed, the SmartLink (ISF) is based on a flexible architecture, offering support for the multiple environments. Example integration scenarios include:

- Enterprise Application Integration (EAI)
- Message Oriented Middleware (MOMs) (Java Message Service)
- Direct connection to legacy APIs

The following are possible integration architectures:

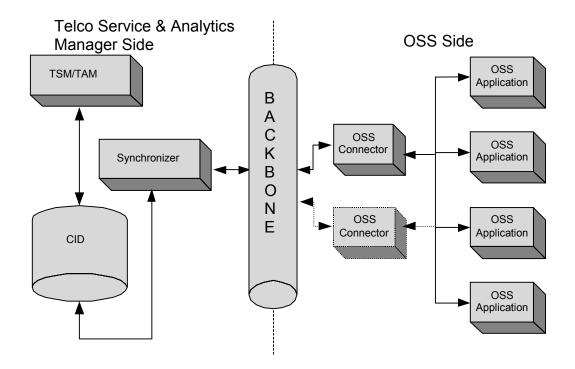
- Connecting to OSS Applications Systems Through an EAI Product
- Connecting to OSS Applications Systems Through an OSS Connector
- Connecting to OSS Applications Directly Through a MOM

Connecting to OSS Applications Systems Through an EAI Product



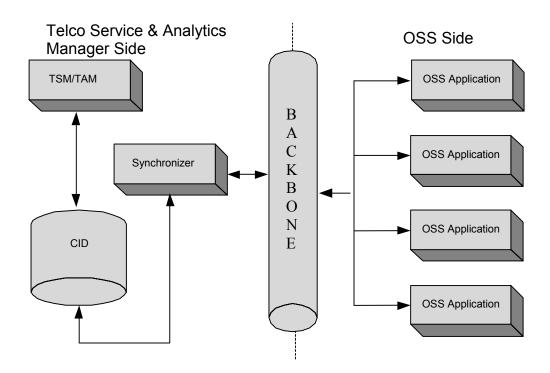
ARCHITECTURE WITH AN EAI PRODUCT	
When to use	EAI product already used by the OSS system
	TSM model can be imported into EAI product repository
	TSM and OSS applications integrated through the EAI product
Benefits	Leverage EAI power (make use of workflow, transaction and process management features)
	Scalable
	Recommended solution

Connecting to OSS Applications Systems Through an OSS Connector



ARCHITECTURE WITH A CONNECTOR ON THE OSS SIDE	
WHEN TO	No EAI product already being used by the OSS system
USE	OSS applications provide exposed Java APIs
	Middleware backbone available
Benefits	Scalable
	Ready to go solution not requiring any EAI product

Connecting to OSS Applications Directly Through a MOM



ARCHITECTURE WITHOUT THE CONNECTOR ON THE OSS SIDE	
When to use	The OSS systems already know how to interact with the middleware backbone
Benefits	Integration with existing middleware backbones

CHAPTER 2

Installing the SmartLink (ISF)

In This Section

Installing the SmartLink (ISF)	32
Uninstalling	33

Installing the SmartLink (ISF)

To install the SmartLink (ISF) components

- 1 Run the installer application for your platform that you downloaded from edocs Customer Central or that is on the Telco Service & Analytics Manager CD-ROM. The Installer appears.
 - AIX: tsm_aix.bin
 - HPUX: tsm_hpux.bin
 - Solaris: tsm_solaris.bin
 - Windows: tsm_windows.exe
- **2** Click *Next*. The License Agreement window appears.
- **3** Read and accept the license agreement then click *Next*. The Location window appears.
- **4** Enter the Telco Service & Analytics Manager home directory then click *Next*. The Installation Type window appears.
- **5** Select *Custom Installation*. The Product Components window appears.
- **6** Select the following components and all of their subcomponents:
 - Synchronization Framework
 - ISF Message Reference
 - OSS/BSS Connectors
- 7 Click Next.
- **8** Follow the on screen instructions. When finished, the installer displays a message.

To install development tools

- 1 Run the installer *composer_windows.exe* that you downloaded from edocs Customer Central or that is on the Telco Service & Analytics Manager CD-ROM (the tools only run on Windows). The Configuration and Deployment Toolkit Installer appears.
- 2 Click Next. The License Agreement window appears.
- **3** Read and accept the license agreement then click *Next*. The Location window appears.
- **4** Enter the home directory then click *Next*. The Installation Type window appears.
- **5** The Installation Type window appears.
- **6** Choose one of the following:
 - All Tools
 - Integration Logic Studio

Presentation Logic Studio

- **7** Choose *Next*. The Choose Shortcut window appears.
- **8** Choose the location of the shortcuts then click *Next*. The Pre-Installation Summary window appears.
- **9** Review the summary then do one of the following:
 - If the summary is correct, click Install to begin installation
 - If the summary is not correct of if you want to change a setting, click Previous.

When finished, the installer displays a message.

CHAPTER 3

Before You Start

In This Section

About Developing Connectors	36
About the Principle Components of Connectors	37
Connector Program Files	41
Selecting the Template	43

About Developing Connectors

Before you begin, you need to be familiar with the various components which make up a connector. You also need to understand the default directory structure and the names of different connector program files.

Once you are familiar with connectors and their components, the first step is to choose the type of connector you need to develop for your TSM.

The SmartLink (ISF) comes with the following connectors you use as templates:

- Synchronizer
- OSS Connector

These connectors contain all of the components you need to start designing your own connector.

When using these connectors as templates, developing connectors involves:

- Selecting the template to use
- Copying the template in a new connector configuration directory
- Designing the connector
- Generate the connector runtime files
- Customizing message structure and mapping

About the Principle Components of Connectors

In general, connectors have the following components:

Integration Processes

An Integration Process is the dataflow for requests or messages between a system and the transport layer, or between the transport layer and one or more systems.

An integration process has one or more processors, associated scripts and macros.

They are represented by the following icon:

■■■ IntegrationProcess

Macros

A Macro is a definition of external APIs to be used as commands.

They are represented by the following icon:



Extensions

An extension is a custom element you use to build Integration Processes.

They are represented by the following icon:



Parameters

A parameter is a connector-wide variable you use when setting element properties.

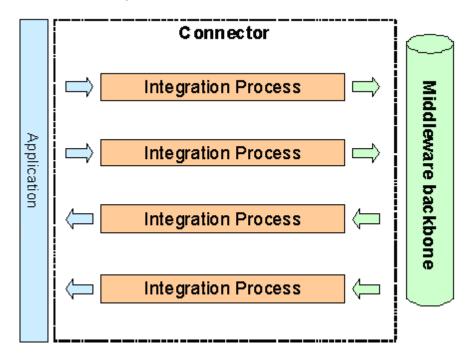
They are represented by the following icon:



Integration Processes

An Integration Process is the dataflow for requests or messages between a system and the transport layer, or between the transport layer and one or more systems. Each connector includes one or more Integration Processes.

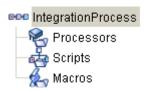
These processes consist of a number of processors connected together in a sequence to handle messages.



An integration process has the following:

- One or more Processors
- One or more Scripts
- One or more Macros

They are represented by the following icon:



The components of the integration process carry out the following basic tasks:

1 Extract Data

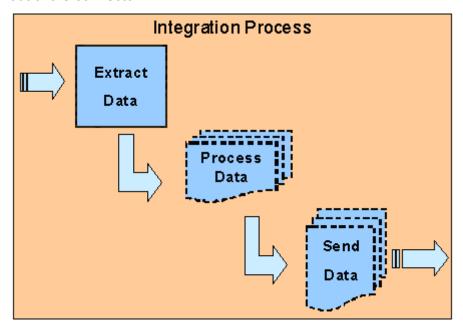
This task is extracting the data the integration process works on. This is a mandatory task and is carried out by a unique processor.

2 Process Data

This task is processing the extracted data. This is also mandatory and has several different components which handle data, forwards data to another processor, generates messages, handles errors, routes messages and so on.

3 Sending Data

This optional tasks is at the end of the processing and involves sending the data out of the connector.



Macros

A Macro element defines the custom APIs that can be called in scripts.

When using Macros to create your own API, you can:

- Wrap external Java code to custom script APIs
- Instantiate external Java classes

For instance, your connector may have to establish a connection with a back end system when the connector starts up and then reuse this connection for all Integration Processes. In this case, you can use a Macro to instantiate the connection then reuse it whenever necessary in any of your Integration Processes.

A macro can be one of the following:

Connector Macro

These macros can be used by all of the processors. They are located under <connector home>/Macros.

Integration Process

These macros can only be used by the elements of the Integration Process. They are located under <connector home>/<integration process>/Macros.

Scripts

This element points to an XML file. This XML file is a Script.

A Script is a sequence of commands or operations, which are executed by a Script Engine Processor. Each command actually calls an API, either defined within the SmartLink (ISF) or registered as an external Macro.

A number of core Script features include a Switch command which allows the script to branch conditionally on the results of any variable, providing "If-Then-Else" functionality within the script. Scripts can also execute sub-script components

They are represented by the following icon:



Because of the power and flexibility of Scripts, they are used for a number of functions:

- Updating data in the CID
- Routing messages within an Integration Process
- Calling external APIs for data processing

Connector Program Files

A connector and its components are built and configured using the following files.

Connector Design File

This file is the design file used by the ILS tool to build and configure your connector. The name of this file is the name of the connector and its extension is ilcd.

This file is automatically generated file. Do not manually edit this file.

Connector Runtime Files

These files are generated by the ILS tool and have the same name as the connector design file.

Connector Configuration File

This file is a runtime connector file containing the private and internal configuration settings. The extension is ilcc.

This file is automatically generated file. Do not manually edit this file.

Connector Parameters File

This file is a runtime connector file. It contains the runtime parameters of your connector. The extension is ilcr.

You can edit this file to customize your configuration after deployment.

Macro Files

These files are the macro files.

Script Files

These files are the script files.

Message Schema Files

These XML files contain the structure and mapping of messages. Each type of message has a corresponding message schema file.

Configuration Files

These files are for configuring various parts of the connector. These files configure the database connection, system logger settings, and so on.

By default, they are installed in:

- <connector_home>/config/connectors/connectortemplate for the OSS connector
- <connector_home>/config/synchronizers/synchronizer for the
 Synchronizer

where <connector home> is the location where the connectors are installed.

The default connector directories:

DIRECTORY		CONTENTS
<connector_name></connector_name>		Contains the following files:
		Connector design fileConnector runtime files
	/macro	Contains the connector macro files
	/nmycfg/util	Contains the configuration files
	/schema_map	Contains the mapping files for messages
	/script	Contains the connector script files

Selecting the Template

Selecting the template you want to use to create your connector depends on the role it plays in integration your TSM. There are two connector templates you can choose from. These templates correspond to the two basic types of connectors.

Once you have decided which type of connector you want to build, you do the following:

- 1 Create a copy of the installed template files. You copy the entire contents of the folder which contains the template.
- 2 Rename the directory and files

Once you have created a copy of the template, you are ready to begin working with the ILS tool to design and generate your connector.

To create a copy of a template

- For a Synchronizer connector, create a copy of the entire contents of <connector home>/config/synchronizers/synchronizer
- For an OSS connector, create a copy of the entire contents of <connector_home>/config/connectors/connectortemplate
 where <connector home> is the location where the connectors are installed.

To rename the template

To change the name of your template, you rename:

- The directory containing the template.
- The ilcd connector design file.
- The connector name

You can change the connector name when you open it using the ILS.

The directory and configuration file must have the same name. If they do not, the connector will not run.

For instance, if you are creating a Synchronizer connector called MySynch, you copy the contents of the <isf_home>/config/synchronizers/synchronizer in <isf_home>/config/synchronizers/MySynch. You then rename the synchronizer.ilcd file to MySynch.ilcd.

CHAPTER 4

Designing Your Connector

In This Section

Overview of Designing Connectors	46
About the Synchronizer Template	47
About the OSS Connector Template	
Overview of Modifying Existing Integration Processes	
Overview of Creating Integration Processes	

Overview of Designing Connectors

Designing your connector using the templates involves:

- Modifying existing integration processes
 - Adjusting the configuration of existing processors to suit your specific needs
 - Adding specific message content or type processing
 - Adding processors in order to perform specific processing on messages
 - Adding message routing features
- Creating new integration processes
 - Creating new message queues
 - Creating new message processing

You use the Integration Logic Studio (ILS) to modify and create integration processes.

About the Synchronizer Template

This connector is a pre-configured, ready-to-run connector. This connector does the following:

Extracts the requests from the CID.

For each extracted request:

- Generates the corresponding XML message
- Sends the XML message over the middleware backbone

The CID Requests Generation Integration Process manages these actions.

- Manages incoming messages from the back end (systems or other connectors)
 - Extracts the incoming XML message from the middleware backbone
 - Parses the message
 - Creates the corresponding CID business objects
 - Updates the CID using the CID business objects

The CID Update Integration Process manages these actions.

About the OSS Connector Template

This connector template is a pre-configured OSS connector. This connector does the following:

- Manages the incoming messages from an TSM connector (Synchronizer)
- Extracts the incoming message from the middleware backbone
- Parses the XML message
- Calls the Script APIs to update your back end system with the information in the message
- Using the reply of the back end system, builds the corresponding reply message
 - Generates the corresponding XML message
 - Sends the message over the middleware backbone

The Inbound_Message_Handling Integration Process handles the core actions of this connector.

This connector must be generated by the Integration Logic Studio before you can run it. This template handles messages but does not process the data they contain.

Overview of Modifying Existing Integration Processes

The integration of your TSM may require you to modify the default configuration and behavior of existing integration processes.

You can modify:

- Message queue parameters
- Request Extraction parameters
- Transmapper Message type lists
- Retry parameters

If your messages have custom fields or when you use custom message types, you may have to modify the following:

- Transmapper Message type lists
- Message mapping and structure definition files
- Scripts which process a specific message type

If you need to perform specific processing on messages, you may have to modify the following:

- Message Queues
- Retry and error handling components

If you need to add message routing features, you may need to modify the following:

- Message envelope based routing through message broadcasters and message type filters
- Message content based routing through scripts

Overview of Creating Integration Processes

If the template does not meet all of your needs or if your TSM requires more specific handling of messages, you create integration processes in your connector .

Creating integration processes involves:

- 1 Creating a new integration process and entering its name
- 2 Create a single processor in charge of extracting your data, either through a message inbound queue or an request queue agent or any extension you may have created.
- 3 Add as many processors you need in order to perform the necessary processing on your data and connect them to create your data flow.

If necessary, add one or several processors in charge of sending your data to a middleware using a message outbound queue or any extension you may have created.

CHAPTER 5

Building Your Connector

In This Section

Working with Connectors	52
Working with Integration Processes	
Working with Processors	
Working with Scripts	94
Working with Macros	
Working with Extensions	113
Working with Parameters	

Working with Connectors

About Connectors

A connector is part of the SmartLink (ISF) that is responsible for moving data between applications and transport layers.

There are two types of connectors:

Synchronizers

They are responsible for data exchanged between your TSM and the transport layer

- BSS/OSS Connectors
- They are responsible for data exchanged between the BSS/OSS applications and the transport layer

Each connector includes the following components:

Integration Processes

An Integration Process is the dataflow for requests or messages between a system and the transport layer, or between the transport layer and one or more systems.

One or more Integration Processes may be declared as being entry points of messages.

Macros

A Macro is a script to define external APIs to be defined and used as commands.

Extensions

An extension is a custom element you use to build Integration Processes.

Parameters

A parameter is a connector-wide variable you use when setting element properties.

To create connectors

- 1 Choose File > New. The Select a connector template dialog box appears
- **2** Browse and select the directory containing the connector to use as a template. The *Select your connector configuration home directory* dialog box opens.
- **3** Browse and select the parent directory of your connector home directory and files then choose *Select*. The *Create new Connector* dialog box appears.
- **4** Enter the name of your connector. This name is used for:
 - The directory containing the connector files
 - The name of the connector design file
- **5** Select *OK*. The Integration Logic Studio creates the connector and its corresponding files in the specified directory. When finished, the connector appears in the Connector explorer and a message appears in the message pane.

To open connectors

- **6** Choose *File > Open*. The *Open Connector* dialog box opens.
- **7** Select the connector file to open.
- **8** Choose *Open*. The connector opens in the Connector Explorer.

To validate connectors

Choose Connector > Check connector. The ILS validates the connector.

After verification, the ILS displays the result in the message pane. If your connector is invalid or if a problem occurs during verification, the ILS displays messages to help you resolve the problem.

To close connectors

1 Choose *File > Close*. The connector closes. If any changes have not been saved, you are prompted to save or discard the changes.

To save connectors

- **1** Do one of the following:
 - Choose File > Save to save the connector.
 - Choose File > Save As to save the connector with a different name.

To generate documentation

- 1 Choose Connector > Generate documentation. The Export Documentation dialog box opens.
- **2** Select the location and enter the name of the HTML file to generate.
- **3** Choose *Save*. The ILS generates the contents of the connector in an easy-to-read HTML file. When finished, the ILS displays a message in the message pane.

Working with Integration Processes

A connector includes one or more Integration Processes.

An Integration Process is the dataflow for requests or messages between a system and the transport layer, or between the transport layer and one or more systems.

Each Integration Process consists of a number of Processors connected together in a specific sequence. An Integration Process may also be declared as being an entry point of messages in the connector.

Working with Integration Processes involves:

- Creating Integration Processes
- Copying Integration Processes
- Opening Integration Processes
- Renaming Integration Processes
- Deleting Integration Processes

To create Integration Processes

- **1** Open the connector.
- **2** In the Connector Explorer, expand *Integration Processes*.
- 3 Right click *Integration Processes* then choose *New*. The *Create New Integration Process* dialog box appears.
- **4** Enter the name of the integration process then choose OK. The new Integration Process appears.

To open Integration Processes

- **1** Open the connector.
- 2 In the Connector Explorer, expand *Integration Processes*.
- **3** Double-click the Integration Process. The Integration Process appears in the Workspace.

To close Integration Processes

- 1 On the Workspace, select the tab of the Integration Process to close. The Integration Process appears in the Workspace.
- **2** Right click the tab then choose *Close*. The Integration Process disappears from the Workspace.

To refresh the contents of an Integration Process

- 1 On the Workspace, select the tab of the Integration Process to refresh. The Integration Process appears in the Workspace.
- **2** Right click the tab then choose *Reload*. The Integration Process view in the Workspace refreshes and displays all of the components which currently make up the process.

To copy an Integration Process

- **1** Open the connector.
- 2 In the Connector Explorer, expand *Integration Processes*.
- 3 Select the Integration Process to duplicate.
- **4** Right click the Integration Process then choose Copy.
- **5** Right click the location under *Integration Processes* where you want to place the process then choose *Paste*. The *Create New Integration Process* dialog box appears.
- **6** Enter the name of the integration process then choose OK. The copy of the Integration Process appears with the name you entered.

To rename Integration Processes

- 1 Open the connector.
- 2 In the Connector Explorer, expand *Integration Processes*.
- **3** Select the Integration Process to rename.
- 4 Slowly double-click the name. The name changes to a text field.
- **5** Enter the new name of the Integration Process.

To delete an Integration Process

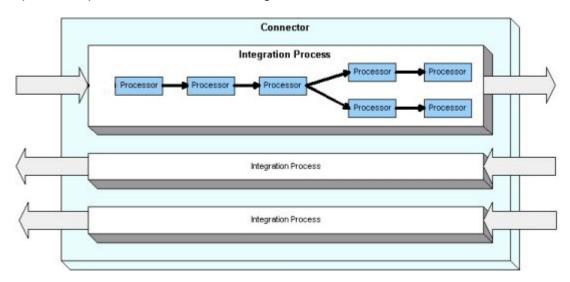
- 1 In the Connector Browser, find the Integration Process to remove.
- **2** Right-click the Integration Process then select *Delete*. A confirmation dialog box appears.
- **3** Choose one of the following:
 - Yes to permanently remove the element from your configuration
 - No to cancel

Working with Processors

About Processors

An Integration Process is the dataflow for requests or messages between a system and the transport layer, or between the transport layer and one or more systems.

Each Integration Process consists of a number of Processors connected together in a specific sequence as shown in this diagram:



There are several different types of processors you can use to build your Integration Process. They include:

- Request Queue Processors
 - These processors handle the extraction of requests from the CID.
- Transmapper Processors
 - These processors handle transforming Java objects used for interaction with systems (your TSM and BSS/OSS) and XML strings (used for message transport)
- Utilities and Error Processors
 - These processors are a collection of processors to assist with common integration workflow issues, such as error handling, retries, wait states, and so on.
- Routing Processors
 - These processors handle complex dataflows where data does not simply flow from one processor to the next.

About Request Queue Processors

The processors in this category are for managing messages extracted from the request queue. The request queue is the table in the CID which holds the requests of the TSM.

The available Request Queue Processors include:

Request Queue Agent

Request Queue Agents periodically collect requests from the CID. As this processor reads information in the CID, it is used in Synchronizer connectors. The frequency at which requests are collected can be configured anywhere from every second (for service subscriptions, handset theft etc) to a matter of hours (address changes etc). The Agent Processor can be configured to retrieve only specific types of request. This effectively determines which Integration Process, or even which Synchronizer, handles which requests.

There can only be one Agent Processor per connector. However, the Agent Processor can be configured to generate multiple threads to increase throughput, and a number of other parameters can be defined to improve the overall messaging performance between your TSM and the BSS/OSS platforms.

These parameters include:

- Number of requests processed per cycle
- Weighting of request types processed per cycle
- Reduced repeat cycle time for high priority events

About Transmapper Processors

The processors in this category are used to transform Java objects (used for interaction with systems – TSM and BSS/OSS) and XML strings (used for message transport).

For example, the connector receives a message in the form of a BLM object. This message needs to be transformed into an XML format the connector can work with. The BLM/XML Transmapper handles converting a BLM object into XML.

Transmappers may also have associated files that specify the structure of the output message. They may also have files that manage the mapping of the input and output messages. The BLM/XML Transmapper has both of these files to define the structure of the XML and the mapping of BLM objects to XML elements.

BLM/XML Transmapper

The BLM/XML Transmapper Processor transforms a Request object into an XML string. This Processor will normally follow immediately after the Request Queue Agent Processor, transforming the minimal information held in the Request object into a complete XML message providing all data required by the target system(s). The specification for this information both in terms of the XML schema and the CSS Engine APIs required to extract the data from the CID, are defined in the Message Schema Reference.

Java/XML Transmapper

The Java/XML Transmapper Processor is the more generic form of the BLM/XML Transmapper. The Java/XML Transmapper Processor transforms any Java object into an XML string. This processor will often be used within a BSS/OSS Connector to transform BSS/OSS objects into an XML message for delivery to the TSM.

CID Objects Builder

The CID Objects Builder Processor transforms a Java hash table into Data Access Layer (DAL) objects, for use in updating the CID. Note that the Processor does not actually update the CID. Updating the CID is performed by the Script Engine.

MX XML Parser

The XML Parser Processor transforms an XML string into a Java hash table.

XSL Transmapper

The XSL Transmapper Processor applies XSL stylesheets to the XML. This processor is often used to reformat or parse information from XML files before processing.

About Router Processors

The processors in this category are for broadcasting messages and managing message flows. Routing Processors are used to create more complex dataflows, where data does not simply flow from one processor to the next.

In general, Messages are broadcast to the queues and the queue has an associated filter to determine whether or not the queue accepts the message. There are processors in this category to manage the broadcasting and filtering of messages for a single queue or multiple queue configurations.

- Message Boradcaster
- Multiqueue Message Broadcaster
- Wessage Type Filter
- Wultiqueue Message Filter

The Message Broadcaster Processor and Message Type Filter Processors work together to enable simple branching of dataflows. The Broadcaster Processor broadcasts all messages to a number of Message Type Filter Processors. Each Filter Processor is configured to accept specific message types only, so that, for example, trouble ticket messages are delivered to one outbound queue, while new service requests are delivered to another. The SmartLink (ISF) can therefore support simultaneous message delivery to multiple BSS/OSS components. The initial request can be converted to a single message, which is delivered to multiple components – or additionally transmapped into multiple messages for the different components.

Script Engine

The Script Engine Processor can create very complex sequence of APIs, by running a pre-defined script to determine where each message and/or message type should be routed. As each script consists of a series of customizable command functions or operations, a script can for example, look up the end-user affected by a message, evaluate their rank within the organization (using an external system), and route messages for higher-ranking users towards a fast-track back-office process. Another example might involve calling an external credit-scoring API to help determine the different systems required to provision a new service.

Scripts and script engines have another very specific use. A script is executed to update the CID when a message is received by your TSM. Using the objects created by the CID Objects Builder Processor, the Script Engine calls core APIs (identified by the script) to write the new data into the CID.

About Message Queue Processors

The processors in this category are for message queues. A message queue is a temporary storage location where messages are held.

The types of queues are:

The processors include:

- Inbound
 These queues are for messages that the connector receives for processing.
- Outbound
 These queues are for messages that the connector has processed.
- Persistence

These queues are for messages that have been saved in a database.

-

- Batch
 - 📲 Inbound Batch Queue
- Email
 - Inbound Email Queue
 - Outbound Email Queue
- File
 - Inbound File Queue
 - Properties Outbound File Queue
- JMS
 - Inbound JMS Queue
 - Outbound JMS Queue
- JMS Topic Queue
 - Inbound JMS Topic Queue
 - Outbound JMS Topic Queue
- Memory
 - Inbound Memory Queue
 - Dutbound Memory Queue
- Socket
 - Inbound Socket Queue
 - Properties Outbound Socket Queue

Persistence



ી Database Polling Processor

About Error Processors

The processors in this category are for handling errors. When processing messages, errors may occur and your connector needs to be able to manage them.

For instance, if an error occurs when processing a message, you can configure your connector to retry the message. You can also specify the conversion of an error into another in order to customize your error handling or trigger other events.

Retry Handler

This processor manages the resending of messages. If an error occurs when processing a message or if a connection is lost, this processor tries to send the message to a specified processor again a specified number of times before logging an error.

Message Retry Counter

This processor manages the resending of messages. If an error occurs when processing a message or if a connection is lost, this processor tries to send the message again a specified number of times before logging an error.

X Error Handler

This process handles errors.

Exception Converter

This processor converts Java exception errors.

Z CID Update Error Converter

This processor converts error messages when an error occurs when updating the CID and updates the CID accordingly (in the reason field of the request queue table)

About Utility Processors

A set of processor types which are provided to assist with common integration workflow issues.

Pause

This processor is simply a pause in the processing of messages. It receives a message and waits the specified amount of time before sending it to the next processor.

Encoding Converter

This processor handles assigning character encoding for messages and XML files. By default, the messages and XML files are in UTF-8. This processor can reassign character encoding in order to use the required encoding for BSS/OSS systems.

Macro Runner

This processor handles running macros. It launches the specified macro at this point in the integration process.

DOM Builder (XML/DOM)

This processor converts a XML file into a DOM document.

DOM Serializer (DOM/XML)

This processor converts a DOM document into an XML file.

XML Validator

This processor handles validating XML messages. This Validator uses XML schemas.

Message Counter

This processor acts as a counter in a connector. It counts the messages and sets some variables used when building Integration Processes.

Working with Processors

When working with the different processors, some of the actions you have to carry out are the same for all processors.

Common actions include:

- Creating Processors
- Editing Processor properties
- Renaming Processors
- Deleting Processors

To create Processors

- **1** Open the Integration Process.
- 2 Right-click *Processors* then select *Create New Processor*. The *Choose a Processor Type* dialog box appears.
- **3** Expand the node of the Processor Type to create.
- **4** Select the Processor to create.
- **5** Enter the name of the Processor.
- **6** Select *Confirm*. The new processor appears in the Connector Explorer and the Integration Process displayed in the Workspace.

To edit Processor properties

- 1 Open the Integration Process.
- **1** Expand the *Processors* node.
- 2 Select the Processor to modify.

If the processor has any properties you can set, the properties appear in the Properties pane.

Depending on the processor, you may also be able to add or remove properties. If you can add and remove properties, the *Optional* tab is active.

To add and remove properties

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.

- 3 Select the Processor to modify. The properties appear in the Properties pane. If the processor allows you to add or remove properties, the Optional tab is available.
- **4** Select the Optional tab.
- **5** Do one of the following:
 - Choose Add to add a new property. Depending on the Processor, a dialog box may appear or a new row created in the Properties pane.
 - Select the property then choose Remove to delete the custom property.

To rename Processors

- 1 Open the Integration Process in the Workspace.
- **2** Select the Processor to rename.
- 3 Select the name of the Processor. The name changes to a text field.
- **4** Enter the new name of the processor.

To delete processors

- 1 Open the Integration Process in the Workspace.
- 2 Find the Processor you want to delete.
- **3** Press *Del*. A conformation dialog box appears.
- **4** Do one of the following:
 - Choose OK to confirm your changes. The Processor disappears from the Workspace and a message appears in the message pane.
 - Choose Cancel

Configuring Processors

Configuring Request Queue Processors

Configuring Request Queue Processors involves:

- Specifying the properties of the processor
- Specifying the request types to manage
- Specifying the sequences
- Specifying the filters

Specifying the Request Queue Agent Properties

To specify the Request Queue Agent properties

- **5** Open the *Integration Process*.
- **6** Expand the *Processors* node.
- **7** Select the *Request Queue Agent* processor to modify. The properties appear in the Properties pane.
- **8** Enter the following properties in the Properties pane:

PROPERTY	DESCRIPTION
Number of Threads	Number of threads to allocate
Napping Time	The period of time between polling (in milliseconds)
Looping Time	The period of time between each internal queue filling (in milliseconds)
Extract Synchronous requests only	Boolean to specify if the Request Queue agent extracts only Synchronous requests
Execute once	Boolean to specify if the Request Queue agent runs once. If true, the Request Queue agent extracts and processes the specified number of requests then shuts down.
No. of Requests to Extract	The number of requests to extract from the request queue
Internal Queue Size	The number of requests in the internal queue of extracted requests

Reconnection Delay	The time in milliseconds to wait between attempts to reconnect to the database
Reconnection Retries	The number of times the agent must try to reconnect to the database
BLM User Name	The user name used to authenticate the connector with the BLM
BLM User Password	The associated password

Specifying the request types to manage

To specify request types

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Expand the Request Queue Agent processor to modify.
- 4 Right-click Request Types then select Create New Request Type. The Create New Request Type dialog box appears.
- **5** Enter the name of the Request Type then select *OK*. The Request Type appears in the Connector Explorer.
- **6** Select the new Request Type then enter the following properties in the Properties pane:

PROPERTY	DESCRIPTION
Enabled	Boolean to specify if the message type is enabled or disabled.
Execution Mode	The execution mode for the request type.
	Can be one of the following:
	- Asynchronous
	- Synchronous
	- Asynchronous MultiQueue
BLM Request Type	The Request Type code as declared in the CID.
Message Type	The associated REQ message.

To specify the status code of the Request Type

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Expand the Request Queue Agent processor to modify.
- **4** Expand the Request Type to modify. The *Status Code* node appears.
- **5** Select the *Status Code* node. The properties of this element appear in the Properties pane.
- **6** Select the status values of the following properties:

PROPERTY	DESCRIPTION
Success Status	The message has been sent successfully
Failure Status	The message could not be sent
Transport Retry Status	the send operation is being retried
Transport Failure Status	The send operation for this message has failed
Waiting for ACK Status	Waiting for an Acknowledge message for the request
Acknowledged Status	An Acknowledge message has been received for the request

Specifying the Sequences

To activate sequences

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- 3 Expand the Request Queue Agent processor to modify.
- **4** Select *Sequence*. The properties appear in the Properties pane.
- **5** In the Properties pane, select the value of the property:

PROPERTY	DESCRIPTION
Enabled	Specifies if sequences are enabled.
	Possible values:
	- true
	- false

To specify sequences

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Expand the Request Queue Agent processor to modify.
- **4** Select *Sequence*. The properties of this element appear in the Properties pane.
- **5** In the Properties pane, select *Add*. The *Request* row appears.
- 6 In the new row, enter the name of the Request.
- **7** Repeat as required.

Specifying Filters

To specify filters

- 1 Open the Integration Process.
- 2 Expand the *Processors* node.
- 3 Expand the Request Queue Agent processor to modify.
- **4** Right-click the *Filters* then select *Create New Filter*. A list of possible filters appears.
- **5** Select one of the following filters:

FILTER	DESCRIPTION
BLM Request Type	The type of the request
Request Status Code	The status code of the request
Request Generator	The Login ID of the user who generated the request
Impacted Object	The object impacted by the request
Impacted Object Type	The object type of the object impacted by the request
Request Creation Date	The creation date of the request
Organization ID	The Organization ID of the level impacted by the request
Starting Date	The start date of the request
Ending Date	The end date of the request
Order By Date	The date to use for sorting

The new Filter appears in the Connector Explorer.

6 Select the new Filter then enter the following properties in the Properties pane:

PROPERTY	DESCRIPTION
Enabled	Specifies if the filter is active or inactive.
	Possible values:
	- true
	- false

7 If required, add a custom property.

In the Properties pane, do the following:

- 7. Click Add. A new property called Request appears.
- 8. Enter the value of this new property.

To activate filters

- 1 Open the *Integration Process*.
- 2 Expand the *Processors* node.
- 3 Expand the Request Queue Agent processor to modify.
- **4** Select *Filter*. The properties appear in the Properties pane.
- **5** In the Properties pane, select the value of the property:

PROPERTY	DESCRIPTION
Enabled	Specifies if filters are enabled.
	Possible values:
	- true
	- false

To set the values of the filter

- 1 Open the Integration Process.
- 2 Expand the *Processors* node.
- **3** Expand the *Request Queue Agent* processor to modify.
- **4** Select *Filter*. The properties appear in the Properties pane.
- **5** In the Properties pane, select *Add*. The *Value* row appears.
- 6 In the new row, enter the value of the filter.
- 7 Repeat as required.

Configuring Transmapper Processors

Configuring Transmapper Processors involves:

- Specifying the request types to manage
- Specifying the message types to manage
- Specifying the location of the configuration files

Configuring BLM XML Transmappers

Configuring BLM/XML Transmappers involves:

- Specifying the request types
- Specifying the configuration files for each request type

To specify request types

- 1 Open the *Integration Process*.
- **2** Expand the *Processors* node.
- 3 Expand the BLM/XML Transmapper to modify.
- **4** Right-click *Request Types* then select *Create New Request Type*. The *Create New Request Type* dialog box appears.

- **5** Enter the name of the Request Type then select OK. The Request Type appears in the Connector Explorer.
- **6** Select the new Request Type then enter the following properties in the Properties pane:

PROPERTY	DESCRIPTION
Structure definition file path	Full path of the XML file containing the definition of the message structure
Mapping definition file path	Full path of the XML file containing the definition of the mapping of BLM content to XML.

Configuring CID Object Builder Transmappers

Configuring CID Object Builder Transmappers involves:

- Specifying the path of the schema definition files
- Specifying the message types

To specify message types

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- **3** Select the *CID Object Builder* transmapper to modify. The properties appear in the Properties pane.
- **4** In the Properties pane, select *Add*. A *Message type* row appears.
- **5** In the new row, select the message type.
- **6** Repeat steps 4 and 5 as necessary.

To specify the location of schema definition files

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- **3** Select the *CID Object Builder* transmapper to modify. The properties appear in the Properties pane.
- 4 In the Properties pane, select Schema def path.
- **5** Enter the location of the Schema definition files to use.

Configuring XML Parser Transmappers

Configuring XML Parser Transmappers involves:

- Specifying the Message Types
- Specifying the configuration files for each Message Type

To specify request types

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- **3** Expand the XML Parser to modify.
- **4** Right-click Request Types then select Create New Request Type. The Create New Request Type dialog box appears.
- **5** Enter the name of the Request Type then select OK. The Request Type appears in the Connector Explorer.
- **6** Select the new Request Type then enter the following properties in the Properties pane:

PROPERTY	DESCRIPTION
Structure definition file path	Full path of the XML file containing the definition of the message structure

To add Request Type Properties

- 1 Open the *Integration Process*.
- **2** Expand the *Processors* node.
- **3** Expand the XML Parser processor to modify.
- **4** Select the Request Type to modify. The properties of this element appear in the Properties pane.
- 5 In the Properties pane, select *Add*. The *Mapping definition file path* row appears.
- 6 In the new row, enter the full path of the XML file containing the definition of the mapping of Java to XML.

Configuring Java/XML Transmappers

Configuring Java/XML Transmappers involves:

- Specifying the request types
- Specifying the configuration files for each request type

To specify request types

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- **3** Expand the Java/XML Transmapper to modify.
- 4 Right-click Request Types then select Create New Request Type. The Create New Request Type dialog box appears.
- **5** Enter the name of the Request Type then select *OK*. The Request Type appears in the Connector Explorer.
- **6** Select the new Request Type then enter the following properties in the Properties pane:

PROPERTY	DESCRIPTION
Structure definition file path	Full path of the XML file containing the definition of the message structure
Mapping definition file path	Full path of the XML file containing the definition of the mapping of BLM content to XML.

Configuring XSL Transmappers

Configuring XSL Transmappers involves:

- Specifying the XSL file to use
- Specifying the message to create

To configure the XSL Transmapper

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- 3 Select the XSL Transmapper to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Buffered Output	Specifies if the output is buffered.
	Possible values:
	- true
	- false
Output Mode	Specifies if the resulting transformations are sent as a single message or each transformation has its own message.
	Possible values:
	- single
	- multiple
Output Format	Specifies the format of the output.
	Possible values:
	- string
	- DOM

To specify

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Expand the XSL Transmapper to modify.
- **4** Right-click *Transformations* then select *Create New Transformations*. The *Create new Transformation* dialog box appears.
- **5** Enter the name of the transformation then select *OK*. The Transformation appears in the Connector Explorer.
- **6** Select the new Transformation then enter the following properties in the Properties pane:

PROPERTY	DESCRIPTION
Stylesheet	Full path of the XSL stylesheet containing the styles to apply.
Output Message Type	The output message type.

Configuring Router Processors

Configuring Transmapper Processors involves:

- Specifying the scripts of the Script Processor
- Specifying the message filters
- Specifying the location of the configuration files

Configuring Script Engines

Configuring Script Engine Processors involves:

Specifying the scripts

To specify the scripts

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Select the Script Engine to modify.
- 4 In the Properties pane, select *Add*. A new Script row appears.
- 5 Select the script from the drop-down list.
- 6 Repeat as necessary.

Configuring Message Broadcasters

Configuring Message Broadcasters involves:

Specifying if the message is duplicated

To specify duplicate flows

- 1 Open the Integration Process.
- **1** Expand the *Processors* node.
- 2 Select the Message Broadcaster to modify.
- 3 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Duplicate Flows	Boolean. When set to true, the Message Broadcaster duplicates the message before sending it.

Configuring Message Type Filters

Configuring Message Type Filters involves:

Specifying the filters

To specify the filters

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Select the Message Type Filter to modify.
- **4** In the Properties pane, select *Add*. A new Filter name row appears.
- **5** Enter the Message type name.
- 6 Repeat as necessary.

Configuring MultiQueue Message Filters

Configuring MultiQueue Message Type Filters involves:

- Specifying the filters
- Specifying the filter properties

To specify the filters

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- **3** Select the *MultiQueue Message Type Filter* to modify.
- 4 In the Properties pane, select *Add*. A new Filter name row appears.
- **5** Enter the Message type name.
- **6** Repeat as necessary.

To specify the filter properties

- 1 Open the Integration Process.
- 2 Expand the *Processors* node.
- **3** Expand the *MultiQueue Message Type Filter* to modify.
- **4** Select the filter to modify. The properties of this element appear in the Properties pane.
- **5** In the Properties pane, enter the following properties:

PROPERTY	VALUE
acknowledge	Specifies if the filter needs to send an ACK message

Configuring MultiQueue Message Broadcasters

Configuring MultiQueue Message Broadcasters involves:

- Specifying the Database Driver
- Specifying the Database URL
- Specifying the Database User Name
- Specifying the Database Password

To specify the database properties

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Select the MultiQueue Message Broadcasters to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Database Driver	The name of the database driver to use
Database URL	The location of the CID database
Database User Name	The user name to connect to the database
Database Password	The associated password

Configuring Inbound Message Queue Processors

To configure Inbound Batch Queue Processors

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Select the Inbound Batch Queue Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Directory Name	Full path of the directory to scan for the files
Prefix	Prefix of file list names
Message Type	Message type to create
Msg Count Variable	Variable to use with the Message Counter Processor
Napping Time	Time between polling (in milliseconds)
Number of threads	Number of threads to allocate
Java priority	JVM priority of the threads

To specify the list of files to process

1 Create a new XML file with the same structure as this sample:

- 2 In the <directory> element, specify the full path of the directory holding your legacy invoice files.
- **3** For each invoice file to process, under <files> specify the name of you invoice file using the syntax:

```
<file>invoices_filename.xml</file>
```

This path must be relative to the path specified in the <directory> element.

- **4** For each invoice file specified in your batch list, you can specify additional parameters:
 - These parameters will be attached to each SmartLink (ISF) message corresponding to an invoice file.
 - These parameters can be queried using the Message.getUserValue Script API.

Additionally, a parameter named filename is automatically inserted with the value set to the full file path.

To specify additional parameters, use the syntax as shown in this example.

The following example of an invoice list contains two parameters:

To configure Inbound Email Queue Processors

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Select the Inbound Email Queue Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Host name	Hostname of the POP server and the domain (example: myhost. mydomain.com)
User Name	POP account name
User Password	POP account password
Napping time	Time between polling (in milliseconds)
Number of threads	Number of threads to allocate
Java priority	JVM priority of the threads

To configure Inbound File System Queue Processors

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.

- 3 Select the Inbound File System Queue Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Directory name	Path of the file system queue
Queue Size	Max number of messages
Napping time	Time between polling (in milliseconds)
Number of Threads	Number of threads to allocate
Java Priority	JVM priority of the threads

To configure Inbound JMS Queue Processors

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Select the Inbound JMS Queue Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
JMS Queue Name	Queue name obtained by JNDI
Connection Factory	JMS connection factory obtained through JNDI
Transactional Flag	Boolean. If set to true, creates a transaction for each message
Initial context factory	JMS initial connection factory
Provider URL	URL of the initial context factory provider
Napping Time	Time between polling (in milliseconds)
Number of Threads	Number of threads to allocate
Java Priority	JVM priority of the threads

To configure Inbound JMS Topic Queue Processors

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- 3 Select the Inbound JMS Topic Queue Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Connection Factory	JMS connection factory obtained through JNDI
Transactional Flag	Boolean. If set to true, creates a transaction for each message
Initial context factory	Boolean. If set to true, creates a transaction for each message
Provider URL	JMS initial connection factory
Napping time	Time between polling (in milliseconds)
Number of threads	Number of threads to allocate
Java priority	JVM priority of the threads

- **5** Expand the JMS Topic Queue Processor.
- **6** Select the *Topics* node.
- 7 In the Properties pane, select *Add*. A new *Topic Name* row appears.
- **8** Enter the name of the topic.
- **9** Repeat as necessary.

To configure Inbound Memory Queue Processors

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- 3 Select the Inbound Memory Queue Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Queue name	Name of the queue
Number of Threads	Number of threads to allocate
Java Priority	JVM priority of the threads

To configure Inbound Socket Queue Processors

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- **3** Select the Inbound Socket Queue Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Host Name	Host name
Port Number	The TCP/IP socket to connect to
Number of threads	Number of threads to allocate
Java priority	JVM priority of the threads

Configuring Outbound Message Queue Processors

To configure Outbound Email Queue Processors

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- 3 Select the Outbound Email Queue Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Host name	Hostname of the POP server and the domain (example: myhost. mydomain.com)
User Name	POP account name
User Password	POP account password
Recipient	Email address of the recipient (example: user@mydomain.com)
Xmailer	Xmailer

To configure Outbound File System Queue Processors

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- 3 Select the Outbound File System Queue Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Directory Name	Path of the file system queue

To configure Outbound JMS Queue Processors

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Select the Outbound JMS Queue Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
JMS Queue Name	Queue name obtained by JNDI
Connection Factory	JMS connection factory obtained through JNDI
Transactional Flag	Boolean. If set to true, creates a transaction for each message
Initial context factory	Boolean. If set to true, creates a transaction for each message
Provider URL	JMS initial connection factory

To configure Outbound JMS Topic Queue Processors

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- **3** Select the Outbound JMS Topic Queue Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Connection Factory	JMS connection factory obtained through JNDI
Transactional Flag	Boolean. If set to true, creates a transaction for each message
Initial context factory	Boolean. If set to true, creates a transaction for each message
Provider URL	JMS initial connection factory

- **5** Expand the JMS Topic Queue Processor.
- **6** Select the *Topics* node.
- 7 In the Properties pane, select *Add*. A new *Topic Name* row appears.
- **8** Enter the name of the topic.
- **9** Repeat as necessary.

To configure Outbound Memory Queue Processors

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Select the Outbound Memory Queue Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Queue Name	Name of the queue
Time Out	Maximum delay for placing messages in this queue when the Queue Size threshold has been reached (milliseconds)
	-1 for infinite timeout
Queue Size	Maximum number of messages

To configure Outbound Socket Queue Processors

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- 3 Select the Outbound Socket Queue Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Host Name	Host name
Port Number	The TCP/IP socket to connect to

Configuring Persistence Message Queue Processors

To configure Message Cache Polling Processors

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- 3 Select the Message Cache Polling Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
DB User Name	Name to connect to the database
DB User Password	Corresponding password
DB Connection String	Connection string
DB Driver	Database driver
Cache Name	Name of the message cache
Number of threads	Number of threads to allocate
Napping time	Time between polling (in milliseconds)
Max No. Of Messages	Maximum number of messages to extract each time
Custom Extraction SQL	Path to the file containing custom SQL script
User exits class	Class containing user exit code to execute
Unread Messages Only	Boolean. If set to true, extract only unread messages
Save Error Description	Boolean. If set to true, save the description of any error

To configure Database Polling Processors

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- **3** Select the Database Polling Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
DB User Name	Name to connect to the database
DB User Password	Corresponding password
DB Connection String	Connection string
Number of threads	Number of threads to allocate
Napping time	Time between polling (in milliseconds)
Flag Read Line	Boolean. If set to true, flags the row as being read
Delete Row On Success	Boolean. If set to true, deletes the row when the extraction occurs without errors
Save Error Description	Boolean. If set to true, saves the description of any error.
User exits class	Class containing user exit code to execute
Identifier Column Name	Name of the column containing the ID name
Identifier Column Type	Type of the column containing the ID
Selection Query	File name of the code used to select messages
Lock Query	File name of the code used to lock messages
Message Factory Class	Class containing the message factory
Confirm Update	SQL file called after successfully processing the message
Confirm Error	SQL file called after processing of a message fails
Confirm Error With Msg	SQL file called after processing of a message fails and the error description field of the message must be updated

Configuring Error Processors

To configure Retry Handler Processors

- 1 Open the Integration Process.
- 2 Expand the *Processors* node.
- 3 Select the Retry Handler Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Number of Repeats	The number of attempts to send the message
Delay between Repeats	The time between attempts (in milliseconds)

To configure Message Retry Counter Processors

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Select the Message Retry Counter Processors to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Number of Retries	The number of attempts to send the message
Delay	The time between attempts (in milliseconds)

To configure Error Handler Processors

You do not need to configure this Processor.

To configure Exception Converter Processors

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- **3** Select the Exception Converter Processor to modify.
- 4 In the Properties pane, select *Add*. The *Add property* dialog box opens.
- **5** Enter the name of the source exception and the target exception.
- **6** Select OK. The new property appears in the Properties pane.

To configure CID Update Error Handler Processors

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- **3** Select the CID Update Error Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Error string	The error message used tu update the reason column of the REQUEST table in the CID

Configuring Utility Processors

To configure the Pause Processor

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- **3** Select the Pause Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Delay	The time to wait between two actions (in milliseconds)

To configure the Encoding Converter Processor

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- 3 Select the Encoding Converter Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Middleware Encoding	The name of the character set to use for the messages
XML Encoding	The name of the character set to use for the XML files

To configure the Macro Runner Processor

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- 3 Select the Macro Runner Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Macro Namespace	The name of the macro to run when a message reaches this processor

To configure the XML Validator

- 1 Open the *Integration Process*.
- **2** Expand the *Processors* node.
- 3 Select the XML Validator Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Validation Flag	Specifies if the processor is to validate the XML.
	Possible values:
	- true
	- false
Schema File	Full path of the XSD file to use for validation.
Schema Format	The format of the schema
	Possible values:
	- W3C XML Schema ("http://www.w3.org/2001/XMLSchema")
	- Relax NG ("http://relaxng.org/ns/structure/0.9")
	- Relax Core ("http://www.xml.gr.jp/xmlns/relaxCore")
	- Relax Namespace ("http://www.xml.gr.jp/xmlns/relaxNamespace")
	- Trex ("http://www.thaiopensource.com/trex")

To configure the Message Counter Processor

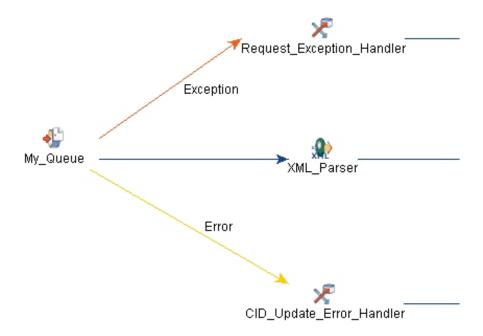
- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- **3** Select the Message Counter Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Msg Count Variable	The name of the variable to use to track the total number of messages that pass through the counter
Success Count Variable	The name of the variable to use to track the number of messages that are successfully passed on to the next processor
Exception Count Variable	The name of the variable to use to track the number of messages that cause exceptions and are not passed on to the next processor

Connecting Processors

The Integration Process is the dataflow for messages and requests. The dataflow consists of a set of processors and the connections between them. Each processor has an input connection and an output connection which trace the path of data through the Integration Process.

In the Workspace, the connections are represented by arrows connecting the processors.



Depending on the connector, the following types of connections can be made:

Send Connection

This connection is the standard connection between processors.

In the Workspace, this connection is represented by a blue arrow.

Error Connection

This connection provides the processor with the initial message content that generated the error. This is the flow of data when an error occurs.

In the Workspace, this connection is represented by a yellow arrow.

Exception Connection

This connection is a connection between a processor and a processor which handles exceptions. This is the flow of data when an exception occurs. A processor with an exception connection can catch exceptions for other processors further on in the data flow.

In the Workspace, this connection is represented by a red arrow.

Working with connections involves:

- Creating connections
- Deleting connections

To connect Processors

- 1 Open the Integration Process in the Workspace.
- **2** Find the Processor you want to connect.
- 3 Click the Processor then drag the pointer to the destination Processor. Depending on the type of destination Processor, the following may occur:
 - An arrow appears connecting the two Processors.
 - If more than one type of connection can be made, a menu appears asking you to select the type of connection.
 - A dialog box appears informing you that you cannot connect the Processors. This occurs when:
 - The Processor is already connected
 - The Processors are not compatible

To disconnect Processors

- 1 Open the Integration Process in the Workspace.
- **2** Find the Processor you want to disconnect.

- 3 Select the arrow connecting the Processor to other another Processor then press *Del*. A conformation dialog box appears.
- 4 Do one of the following:
 - Choose OK to confirm your changes. The arrow disappears and a message appears in the message pane.
 - Choose Cancel

Working with Scripts

A Script is a sequence of commands executed by a Script Engine Processor. Because of the power and flexibility of Scripts, they are used for a number of functions:

- Scripts update data in CID
- Scripts control message-routing decisions within an Integration Process
- Scripts call external APIs for data processing

•

Working with scripts involves:

- Creating the Script in the Integration Process
- Building the sequence
- Declaring the Script in a Script Engine Process

Working with Scripts

- **1** Open the Integration Process.
- 2 Right-click *Scripts* then select *Create New Script*. The *Create new Script* dialog box appears.
- **3** Enter the name of the script.
- **4** Choose OK. The new script appears.

Before you start working with this script, you must create or specify the script file.

To specify a script XML file

After creating your Script, you need specify the associated script file. A script file is an XML file containing the code of your script.

By default, the script files are located in <connector_home>/scripts.

- **1** Open the *Integration Process*.
- **2** Expand the *Scripts* node.
- **3** Right-click the script then choose *Open*. The file selection dialog box opens.
- **4** Browse to the directory for scripts.
- **5** Do one of the following:
 - Enter the full name of the file (<file name>.xml) to create a new script file
 - Select the script file to associate with the script
- **6** Choose *Save*. The script opens in the Workspace.

To open a script

- 1 Open the Integration Process.
- **2** Expand the *Scripts* node.
- 3 Double-click the Script to open. The script opens in the Workspace.

To close a script

- 1 On the Workspace, select the tab of the Script to close. The Script appears in the Workspace.
- **2** Right click the tab then choose *Close*. The Script disappears from the Workspace.

To delete a script

- 1 Open the Integration Process.
- **2** Expand the *Script* node.
- 3 In the Connector Browser, find the Script to remove.
- 4 Right-click the Script then select *Delete*. A confirmation dialog box appears.
- **5** Choose one of the following:
 - Yes to permanently remove the element from your configuration
 - No to cancel

Programming Scripts

You use the Script Editor to create the sequence. This editor displays the sequence of steps in your script as a simple and easy-to-understand flow chart. This not only helps you understand how the script works, it also makes editing and modifying scripts easy.

Programming a script involves:

- Working with Steps
 - Creating Steps
 - Editing Steps
 - Deleting Steps
- Connecting the steps to create the sequence

Working with Steps

The commands in a script call APIs either defined within the SmartLink (ISF) or registered as an external macro.

The sequence of commands are created using the following:

InitStep

This step starts the sequence. Your Script must begin with an Init Step

InvokeStep

The Invoke Step calls the API or external macro.

SwitchStep

The Switch Step determines the next step depending on the result of a test.

CallStep

The Call Step forwards the processing to another Integration Process or Processor

Working with Steps involves:

- Creating steps
- Setting step properties
- Editing steps
- Deleting steps

When you are finished, you connect the steps.

To create steps

- **1** Open the Script in the Workspace.
- **2** Right-click the Workspace and choose *Create New*. A list of steps to create appears.
- **3** Select the type of step to create. The *Create New Step* dialog box opens.
- **4** Enter the name of the step.
- **5** Select *OK*. The new step appears on the Workspace.

To configure Invoke steps

- 1 Open the Script.
- **2** Select the Invoke step to modify.

3 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Method	The macro expression to call
Variable	The name of the variable to store the result in
Log	Optional text message for the logger

To configure Init steps

- 1 Open the Script.
- 2 Select the Init step to modify.
- 3 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Message Type	The type of the message to process with this script.
	To process all message types, leave blank.
Variable	The name of the variable to store the received message in
Dispatcher	Mandatory property. Must be set to DISPATCHER.
Log	Optional text message for the logger

To configure Call steps

- 1 Open the Script.
- 2 Select the Call step to modify.
- 3 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Processor	The name of the processor to call
Message Type	The type of message to send to the specified processor
Variable	The name of the variable containing the message to send
Reply	The name of the variable containing the message to correlate
Result	The name of the variable that contains the message returned by the specified processor
Log	Optional text message for the logger

To configure Switch steps

- 1 Open the Script.
- **2** Select the Switch step to modify.
- 3 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
Condition	Macro expression to evaluate. This value is transmitted by the transition connection.
Log	Optional text message for the logger

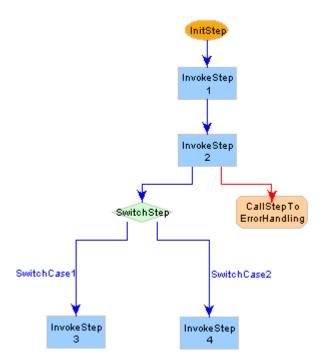
To delete steps

- **1** Open the Script in the Workspace.
- **2** Find the Step you want to delete.
- **3** Press *Del*. A conformation dialog box appears.
- **4** Do one of the following:
 - Choose OK to confirm your changes. The step disappears from the Workspace and a message appears in the message pane.
 - Choose Cancel

Connecting Steps

The script is a sequence of calls.

In the Workspace, the connections are represented by arrows connecting the processors.



The following types of connections can be made:

Transition Connection

This connection is the standard connection between steps.

In the Workspace, this connection is represented by a blue arrow.

Switch Connection

This connection is a standard connection with the value associated with Switch step.

Using a Switch step in your script is to create a point in the processing where the sequence may branch. The Switch step has a Condition which corresponds to the test carried out by the step. The result of this test determines the subsequent step. This result is entered as a property of the connection between the Switch step and the subsequent step.

In the Workspace, this connection is represented by a blue arrow with the value of the decision property displayed next to the arrow.

Exception Connection

This connection is a connection between a step and a step which handles exceptions.

In the Workspace, this connection is represented by a red arrow.

Working with connections involves:

- Creating connections
- Deleting connections

To connect steps

- 1 Open the script in the Workspace.
- **2** Find the step you want to connect.
- 3 Click the step then drag the pointer to the destination step. A menu appears asking you to select the type of connection.
- **4** Select the type of connection. The arrow representing the connection appears.

To create a switch connection

- **1** Open the script in the Workspace.
- **2** Find the Switch step you want to connect.

- 3 Click the Switch step then drag the pointer to the destination step. A menu appears asking you to select the type of connection.
- **4** Select Transition as the type of connection. The arrow representing the connection appears.
- **5** Select the arrow. The properties appear in the Properties pane.
- **6** In the Properties pane, select *Add*. A *Decision* row appears.
- 7 In the new row, enter the value of the Decision property. This is the value returned by the Condition property of the Switch step.

To disconnect steps

- **1** Open the script in the Workspace.
- **2** Find the steps you want to disconnect.
- **3** Select the arrow connecting the steps then press *Del*. A conformation dialog box appears.
- **4** Do one of the following:
 - Choose OK to confirm your changes. The arrow disappears and a message appears in the message pane.
 - Choose Cancel

Working with Macros

Macro statements are logically grouped and stored in a text file. They are loaded and interpreted during connector configuration loading. The Method declaration statements of a macro define custom APIs that can be invoked in scripts.

When using Macros to create your own APIs, you can, for instance:

- Wrap external Java code to custom script APIs
- Instantiate external Java classes
- Perform operations (add, multiply...) on macro variables
- Invoke external java code on macro variables

Macro statements are case sensitive.

Types of Macros

You can create macros which are limited to a specific Integration Process. You can also create create macros which can be used by the all of the processors in your connector.

A macro can be one of the following:

Connector macro

These macros can be used by all of the processors in a connector. They are located under <connector home>/Macros.<connector >/Macros.

Integration Process macro

These macros can only be used by elements of the Integration Process. They are located under

<connector_home>/<integration_process>/Macros.<connector
>/<integration process>/Macros.

Creating and programming both types of macros are the same.

When running the connector, the Macros are loaded in the following order:

- 1 Connector Macros
- 2 Integration Process Macros

To create Macros

- 1 Do one of the following depending on the type of Macro:
 - Open the connector.
 - Open the Integration Process.
- 2 Right-click *Macros* then select *Create New Macro*. The *Create new Macro* dialog box appears.
- **3** Enter the name of the Macro.
- 4 Choose OK. The new Macro appears.

To open Macros

- **1** Do one of the following depending on the type of Macro:
 - Open the connector.
 - Open the Integration Process.
- **2** Expand the *Macros* node.
- 3 Double-click the Macro to open. The Macro opens in the Workspace.

To close a Macro

- 1 On the Workspace, select the tab of the Macro to close. The Macro appears in the Workspace.
- 2 Right click the tab then choose *Close*. The Macro disappears from the Workspace.

To delete a Macro

- **1** Do one of the following depending on the type of Macro:
 - Open the connector.
 - Open the Integration Process.
- **2** Expand the *Macros* node.
- **3** In the Connector Browser, find the Macro to remove.
- 4 Right-click the Macro then select *Delete*. A confirmation dialog box appears.
- **5** Choose one of the following:
 - Yes to permanently remove the element from your configuration
 - No to cancel

This action only removes the macro from your Connector or Integration Process. It does not delete the macro source file.

Specifying the Macro file

After creating your Macro, you need specify the associated macro source file. If the file does not exist, you can create a new macro file using the ILS.

Specifying the macro involves:

- If the file does not exist, creating the macro source file
- If the file exists, declaring the macro source file

To create and specify a macro file

- **1** Do one of the following depending on the type of Macro:
 - Open the connector.
 - Open the Integration Process.
- 2 Expand the Macros node.
- **3** Double-click the Macro. The file selection dialog box opens.
- **4** Browse to the directory to create the macro file in.
- **5** Enter the full name of the file (<file name>.properties) then choose Save.
- **6** The macro file is created and opens in the Workspace.

To specify the Macro

- 1 Do one of the following depending on the type of Macro:
 - Open the connector.
 - Open the Integration Process
- **2** Expand the *Macros* node.
- **3** Select the Macro to modify.
- 4 In the Properties pane, enter the path and file name of the Macro file.

Programming Macros

When you program a Macro, you create the Macro source file.

Programming a Macro involves:

- Creating a source file corresponding to the name entered in the path of your Macro element.
- Editing the Macro

If the right part of an expression contains brackets "xxx()", then it will be evaluated ONLY when referenced by another expression. Otherwise as soon as the expression is parsed it will be evaluated.

Variables

In order to declare a variable and initialize it, you use the following statement:

```
Variable_name=initialization_value
```

The initialization value can be provided by virtually any expression that returns a value. Therefore you can use simple standard java object types, such as:

String

```
Var1="MyString"
```

Character

```
Var1='a'
```

Integer

Var1=10

Boolean

Var1=true

You can also use more complex java objects. For instance, you can use:

Java Vector

```
Var1={1,2,MyString}
```

Java Class Array

```
Var1=java.lang.String[]("aa","bb","cc")
```

Java Hash table

```
MyHash=["key1"=>"value 1","key2"=>"value 2"]
```

Operators

The following operators are defined:

- Arithmetic ('+','-','*','/')
- String ('+')
- Boolean ('&', '||')
- Order relation ('<','>','<=','>=','==','!=')

Static Method Aliases

You can create an alias pointing to a java static method.

To declare a user function alias, use the call statement.

The function syntax is:

FunctionAliasName(Parameters list) = call(String classname, String
MethodSignature)(Parameters list)

The method signature can be defined using one of the following ways:

You specify the method name and the list of your function parameters

```
Alias(P1, P2, ...) = call(classname, MethodName)(P1, P2, ...)
```

 You specify the method name, the list of class method parameters and the list of your function parameters. For each method parameter, you must specify only the java type.

```
Alias(P1, P2,...) = call(classname, MethodNameWithParametersType)(P, 1P2,...)
```

You specify the method name

```
Alias=call(classname, MethodName)
```

 You specify the method name, the list of method parameters. For each method parameter, you must specify only the java type.

```
Alias=call(classname, MethodNameWithParametersType)
```

Specifying the list of the class method parameters is required only when a method has several signatures.

If you specify the list of your function parameters, the referenced custom class is instantiated only when you call the function.

If you do not specify the list of your function parameters, the referenced custom class is instantiated when the macro is loaded. This also implies that it inherits the scope of the macro (connector or limited to the Integration Process.

Class constructors with parameters are not supported with the "call" statement

Your method must be declared as static.

Non-static Method Aliases

You can create an alias pointing to non-static methods of a class.

To declare a non static user function alias, you do the following:

1 Instantiate the class. If required, you can you can specify the parameters that will be passed to your class constructor.

To instantiate a class, use the instantiate statement as follows:

```
ClassAliasName=instantiate(String classname) (Constructor
Parameters)
```

This statement looks for the class constructor and, if found, calls it.

2 Invoke the method of the class. You specify the parameters that are passed to your method.

In order to define an alias to the method of your class, use the following statement:

```
FunctionAlias(P1, P2, ...) = ClassAlias -> MethodName(P1, P2, ...)
```

Method names are case sensitive.

You may need to instantiate your call each time the method is called. In order to do this, use the following statements:

```
ClassAlias() = instantiate(String classname) (Constructor
Parameters)
FunctionAlias(P1, P2,...) = ClassAlias() -> MethodName(P1, P2,...)
```

Class Fields

You can extract class field value.

To extract a class field value, use the getField statement:

thevalue=getField(String classname, String fieldname)

Functions

In order to perform a specific task on one or several objects, you can define a function.

To define a function, you specify the following:

- function name
- arguments
- the function

The syntax is as follows:

```
functionname(list of parameters) = <any macro expression>
```

The function definition itself can consist of any macro statement or method alias.

Namespaces

Use the following statement to declare a namespace:

```
NameSpace_Name {
Other macro statements...
}
```

Use the following syntax to access a previously defined element inside an existing namespace:

```
Namespace Name.OtherElement
```

Samples

Defining Simple Functions

This sample describes the macro statement involved when you need to define a simple function that returns the sum of two numbers.

The Macro statement:

```
sum(N1, N2) = N1 + N2
```

In an Invoke script step, you use the sum(x,y) macro statement to invoke your function.

Wrapping Static Methods

This sample describes the macro statements involved when you need to call a static method from external java code in a script or in a macro.

This example assumes:

- Your java class (MyClass.class) is referenced in your classpath
- Your java class has only one constructor without any parameters
- Your java class contains only one method with the following signature
 MyMethod(String aString, Object anObject)

The macro statement:

```
mymethod(P1, P2) = call("MyClass", "MyMethod")(P1, P2)
```

In an Invoke script step, you use the mymethod(x, y) macro statement to invoke your custom class.

Wrapping Polymorphic Static Methods

This sample describes the macro statements involved when you need to call a polymorphic method from external java code in a script or in a macro.

This example assumes:

- Your java class (MyClass.class) is referenced in your classpath
- Your java class has only one constructor without any parameter
- Your java class contains one method with the following signature
 MyMethod (String aString, Object anObject)

The Macro statement:

```
mymethod(P1, P2) = call("MyClass", "MyMethod(java.lang.String, java.l
ang.Object)")(P1, P2)
```

In an Invoke script step, you use the mymethod(x, y) macro statement to invoke your custom class.

Instantiating Custom Classes

With Constructor Parameters

This sample describes the macro statements involved when you need to instantiate external java classes with constructor parameters.

This example assumes:

- Your java class (MyClass.class) is referenced in your classpath
- Your java class has only one static constructor with one parameter

The Macro statement:

```
myclass=instantiate("MyClass") (parametervalue)
```

In an Invoke script step, you use the myclass macro statement to instantiate your custom class.

With Dynamic Constructor Parameters

This sample describes the macro statements involved when you need to instantiate external java classes with dynamic constructor parameters.

This example assumes:

- Your java class (MyClass.class) is referenced in your classpath
- Your java class has only one static constructor with two parameters

The Macro statement:

```
myclass(P2) = instantiate("MyClass")(P1, P2)
```

When called, the myclass macro statement will call your class MyClass constructor with two parameters:

- The first parameter P1 is set by the myclass (xxx) statement
- The second parameter P2 is set by the macro definition statement itself

Working with Extensions

When building your connector, your environment may require different types of processors that are not available by default. With the ILS, you can create your own processors or inbound and outbound queues that can be reused in all of the Integration Processes of the connector.

These custom processors are referred to as extensions.

Working with extensions involves:

- Creating the extension
- Configuring the extension
- Adding the extension to your Integration Process

To create extensions

- 1 Open the connector.
- **2** Expand the *Extensions* node. The *Inbound Queue* and *Outbound Queue* nodes appear.
- **3** Right-click the node corresponding to the type of extension you want to create then select *Create New Extension*. The *New extension* dialog box appears.
- 4 Enter the name of the extension.
- **5** Choose OK. The new extension appears.

To configure extensions

- **1** Open the connector.
- **2** Expand the *Extensions* node. The *Inbound Queue* and *Outbound Queue* nodes appear.
- **3** Expand the node corresponding to the type of extensions to modify.
- **4** Select the extension to modify. The properties appear in the Properties pane.
- **5** Choose *Add* to add a new property. A new Parameter name row created in the *Properties* pane.
- **6** Enter the value of the Parameter pane.
- 7 Repeat as required.

To add extensions to Integration Processes

- **1** Open the Integration Process.
- **2** Right-click *Processors* then select *Create New Processor*. The *Choose a Processor Type* dialog box appears.

- **3** Expand the *Extensions* node.
- 4 Select the extension to add.
- **5** Enter the name of the extension.
- **6** Select *Confirm*. The new extension appears in the Connector Explorer and the Integration Process displayed in the Workspace.

To delete extensions

- **1** Open the connector.
- **2** Expand the *Extensions* node.
- **3** Find the extension to remove.
- 4 Right-click the extension then select *Delete*. A confirmation dialog box appears.
- **5** Choose one of the following:
 - Yes to permanently remove the element from your configuration
 - No to cancel

Working with Parameters

When working with any connector, even if it is a simple connector with a straightforward workflow, you may find yourself dealing with many different elements and properties. The ILS uses parameters to help organize and simplify setting and changing the value of properties. A parameter is simply a property variable that you can use when setting variables.

For instance, you may have to enter the same property over and over again when configuring your connector. For example, for each message type you have to specify the location of the message structure definition file. If all of your message structure definition files are located in the same directory, you have to enter the same path for each message type. You can replace c:/myconnector/shared/messagedef with the SCHEMA PATH parameter.

If you change the directory, you do not have to open and change the path of each message type. You change the value of the $SCHEMA_PATH$ parameter.

This also helps makes changing directories or other common properties easier.

Parameters can be used to replace values such as:

Paths:

```
path/subpath1/
./path
../../path
```

Strings:

dest@company.com
BlueConnectorBeta

- Boolean values
- Numeric values

For Request Queue Agent Filters, your parameter can have several values. You configure your parameter to have several values and only the declare the parameter in the Request Queue Agent Filter.

Parameter names have the following restrictions:

- No spaces (underscores are allowed)
- Only letters (no numbers or symbols can be used)
- Upper case only

Example of a valid parameter names:

- HOME_PATH
- EMAIL
- CONNECTOR NAME

Working with parameters involves:

- Creating a parameter
- Setting the values
- Using the parameter in property values

To create a parameter

- 1 Open the connector.
- **2** Right-click the Parameters node then select *Create New Parameter*. The *New parameter* dialog box appears.
- **3** Enter the name of the parameter.
- **4** Choose OK. The new parameter appears.

To configure parameters

- **1** Open the connector.
- **2** Expand the *Parameters* node.
- **3** Select the parameter to modify. The properties appear in the Properties pane.
- 4 Enter the following parameters:

PARAMETER	DESCRIPTION
Description	Description of the parameter
Parameter description	Description of the value of the parameter
Туре	The type of parameter.
	Possible values:
	- String
	- Boolean
	- Numeric
Multiplicity	Specifies if the parameter has one or more value.
	Possible values:
	• Simple
	Multiple
Value	The value of the parameter.

- **5** If the parameter is a multiple value parameter:
 - Select Add and enter a new value.
 - Repeat as required.

Multiple value parameters are for Request Queue Agent Processor Filters only.

To use parameters in property values

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- **3** Select the Processor to modify. If the processor has any properties you can set, the properties appear in the Properties pane.
- **4** Enter the name of the parameter as the value of a property. Use the syntax:

\$parameter name\$

Example:

- Parameter name: SCHEMA PATH
- Parameter value: c:/myconnector/shared/messagedef

The value of the DOADDCONTACT message type structure definition file path is:

\$SCHEMA PATH\$/doaddcontact.xml

To use multiple value parameters

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.

- 3 Expand the Request Queue Agent Processor node.
- 4 Expand the Filters and Filter nodes.
- **5** Select the Filter to modify. The properties appear in the Properties pane.
- **6** Enter the parameter name as the value of the property. Use the syntax:

\$parameter_name\$

CHAPTER 6

Working With Connector Extensions

In This Section

About Extensions	120
Defining Custom Processors	121
Validating Custom Processors	122
Using Custom Processors	123
Generating Custom Processors	124
Samples	125

About Extensions

When you are using a custom middleware backbone, or when you need to interact with a custom system, you can use ILS extensions. ILS extensions are custom processors which can be used in different connectors.

For instance, you might need to send messages to an unsupported custom middleware application. Or you might also need to extract data from a back end system in order to send a notification to your TSM connector.

The different custom processors you can create are:

- Inbound Queues
- Outbound Queues

An extension is the logical representation of the custom java code that will be called by the SmartLink (ISF). The properties of your custom processor type are directly used to configure the java instance constructors at run time. Each parameter is directly bound to a parameter of the constructor of your java class.

Defining Custom Processors

Defining your custom processor type allows you to create in integration processes new processors based on your type.

To create a new custom processor:

- 1 Open the connector.
- **2** Expand the *Extensions* node. The *Inbound Queue* and *Outbound Queue* nodes appear.
- 3 Right-click the node corresponding to the type of extension you want to create then select *New > Extension*. The *New extension* dialog box appears.
- 4 Enter the name of the extension.
- **5** Choose OK. The new extension appears.
- 6 If needed, create as many as necessary properties for your processor type. Each property corresponds to a parameter of the java class constructor of your custom processor type java code

Validating Custom Processors

In order to validate a connector using your custom processor type, you have to configure the ILS validation engine to take your custom processor into account.

To configure the validation of a custom processor, do the following:

- 1 Open the <connector_home>/config/custom/custom_processors.xsd
 file.
- 2 Add the required schema declarations for your processor type
- 3 Save your changes.
- 5 Add the required declarations for your processor type
- 6 Save your changes.

The ILS validation engine is now able to validate a connector using your custom processor.

Using Custom Processors

- Open the Integration Process.
- 2 Right-click *Processors* then select *New > Processor*. The *Choose a Processor Type* dialog box appears.
- 3 Expand the Custom Processors node.
- Select the the type of Processor to create.
- Enter the name of the Processor.
- Select *Confirm*. The new processor appears in the Connector Explorer and the Integration Process displayed in the Workspace.
- Specify its properties in the Properties pane.

Generating Custom Processors

To generate a connector using your custom processor type, you have to configure the connector generation tool for your processor type.

To configure the connector generation tool:

- 1 Open the <connector_home>/config/custom/custom_processors.cgf
 file
- **2** Add the required declarations for your processor type.
- 3 Use the syntax:

```
<plugin
name="Your_custom_processor_type">your.class.name</plugin>
```

4 Save your changes.

For example, for an OutboundQueue, you add the following:

```
<root>
  <OutboundQueues>
    <plugin name="Your_custom_processor_type">your.class.name</plugin>
  </outboundQueues>
</root>
```

5 If required, you may need to add your class to the CLASSPATH in the environment settings found in <connector_home>/bin/env.

Samples

Sample Custom Outbound Queue

This sample is the contents of the <code>custom_processors.cgf</code> file and a sample java class that handles the generation of a custom OutboundQueue processor type with two parameters (Parameter1 and Parameter2.)

Sample custom_processors.cgf File

```
<root>
  <OutboundQueues>
<plugin name="MyQueueOutDriver">myqueuedriver_out</plugin>
  </OutboundQueues>
</root>
```

Following is a sample custom_processors.cgf file and a sample java class that handles the generation of a custom processor type (with Category OutboundQueue) with two parameters, Parameter 1 and Parameter2.

Sample java class

```
import com.netonomy.cgf.api.plugin.PluginIF;
import com.netonomy.cgf.api.plugin.PluginQueue;
import com.netonomy.cgf.isf.Processor;
import org.dom4j.Element;
import com.netonomy.cgf.isf.Verbose;
import com.netonomy.cgf.api.plugin.ElementUtil;
import java.util.HashMap;

public class myqueuedriver_out extends PluginQueue {
   public String[] onDriver(Element processor, String connName, String integrationProcessName, String
   processorName, String processorType) throws Exception {
    String[] props = new String[] {
        "PARAMETER1=" + ElementUtil.getPropertyValue(connName, processor, "PARAMETER1", ElementUtil.STRING_TYPE, null,
        ".", true),
        "PARAMETER2=" + ElementUtil.getPropertyValue(connName, processor, "PARAMETER2", ElementUtil.STRING_TYPE, null,
        ".", true);

String[] drivers = new String[] {"myqueuedriver_out=instantiate(\"com.myclasses.myqueuedriverout\")", "DRIVER=
        myqueuedriver_out (properties. PARAMETER1, properties. PARAMETER2,0)");

return ElementUtil.generateConfiguration(props,drivers);
   }
}
```

Sample Custom Inbound Queue

This sample is the contents of the <code>custom_processors.cgf</code> file and a sample java class that handles the generation of a custom InboundQueue processor with two parameters (Parameter 1 and Parameter2.)

Sample custom_processors.cgf File

Sample java class

```
import org.dom4j.Element;
import java.util.Iterator;
import org.dom4j.Attribute;
import java.util.HashMap;
import com.netonomy.cgf.api.plugin.ElementUtil;
import com.netonomy.cgf.api.plugin.PluginAgent;
import java.util.Vector;
import com.netonomy.cgf.isf.Constants;
public class myqueuedriver_in extends PluginAgent {
public String[] onDriver(Element processor, String connName, String integrationProcessName, String
processorName, String processorType) throws Exception {
String[] props = new String[] {
"PARAMETER1=" + ElementUtil.getPropertyValue(connName, processor, "PARAMETER1", ElementUtil.STRING_TYPE, null, "0",true),
" PARAMETER2=" + ElementUtil.getPropertyValue(connName, processor, "PARAMETER2", ElementUtil.INT_TYPE, null,
"0", true) };
String[] drivers = new String[] {
           ElementUtil.getTransitionValue(processor, connName, integrationProcessName, "send",true),
                      ElementUtil.getTransitionValue(processor, connName, integrationProcessName,
"exception", false),
           {\tt ElementUtil.getTransitionValue} \ ({\tt processor}, \ {\tt connName}, \ {\tt integrationProcessName}, \ {\tt "error"}, {\tt false}) \ , \\
           "myqueuedriver_in=instantiate(\"com.myclasses.myqueuedriverin\")",
           "DRIVER=fileQ(properties.PARAMETER1,properties. PARAMETER2)",
           "AGENT=Create\_Queue Agent (properties.NBTHREADS, DRIVER, EXCEPTION, ERROR, properties.PRIORITY)"
          };
         return ElementUtil.generateConfiguration(props,drivers);
public String[] onStart(Element elementProcessor, String connName, String integrationProcessName, String
proccessorName, String processorType) throws Exception
// Nothing to generate
Vector v = new Vector();
// daemon
v.add("STARTWITH=null");
String[] props = new String[v.size()];
for(int i=0;i <v.size();i++){
           props[i] = Constants.TAB1 + (String)v.get(i);
return props;
```

CHAPTER 7

Working with the SmartLink (ISF) Message Cache

In This Section

About the SmartLink (ISF) Message Cache	130
About the Saved SmartLink (ISF) Message Structure	131
About the SmartLink (ISF) Message Cache Script APIs	132
About the SmartLink (ISF) Message Cache Processors	133
Working With the SmartLink (ISF) Message Cache	134
Using the Message Cache to Build a Business Event Queue.	145

About the SmartLink (ISF) Message Cache

You use the message cache to store SmartLink (ISF) messages while awaiting the reply from the OSS.

The message cache allows you to:

- Temporary store of integration messages and related data in order to share them among threads and processes
- Cumulatively and concurrently update stored messages for easier reconciliation

When working with messages, the connector receives a message then calls a back end API to request additional information. This API call is performed in a non blocking manner, which means the connector does not wait for the reply. Depending on your integration architecture, you may need to store the message while awaiting the reply from the backend system. As soon as the connector receives this reply from the back end, it restores the persisted message, does the required task (reconcile, update, and so on). When finished, the connector either persist the message again for another request/reply or sends the updated message then removes the persisted message.

The SmartLink (ISF) message cache has the following components:

- Message Cache database
- Message Cache database administration tool
- Message cache APIs
- Message cache processors

About the Saved SmartLink (ISF) Message Structure

An SmartLink (ISF) message saved in the SmartLink (ISF) message cache has the following:

- Message envelope
- Message body (optional)
- Message parameters (optional and multiple)

The messages have the following characteristics:

- There are no restrictions of the size of the message body. This means you can save entire XML strings in the message body.
- The message envelope fields can be used and set according to your needs.
- The message parameters can be used to store elementary data. Any number of parameters can be attached to a message and hierarchic as well as flat arrays can be used.
- A Message can contain any number of sub messages to create a message tree.

About the SmartLink (ISF) Message Cache Script APIs

You use the SmartLink (ISF) Message Cache Script APIs to do the following:

Extract messages

The extraction criteria include:

- Identifiers: Internal id, external id, correlation id
- Message Envelope: Any field of the envelope (type, status, priority, and so on)
- Custom extraction using custom SQL
- Update messages

Messages can be updated using standard identifiers and any message field can be updated using its corresponding API

Deleting messages

Messages can be deleted using standard identifiers.

- Connecting to the SmartLink (ISF) Message Cache:
 - Opening a connection
 - Closing a connection
- Managing Message Cache transactions:
 - Creating
 - Deleting
 - Commits
 - Rollbacks
 - And so on
- Managing SQL Statements
 - Executing SQL
 - Parsing results
 - And so on

For more information about:

- the SmartLink (ISF) Message Cache Script APIs, refer to the *SmartLink (ISF) Script API* online documentation
- Using scripts in the ILS, refer to Working With Scripts.

About the SmartLink (ISF) Message Cache Processors

You use the following processors in the ILS to build integration processes using the SmartLink (ISF) Message queue:

- Message Cache Poller
 This processor extracts messages from the SmartLink (ISF) message cache.
- Database Poller
 This processor extracts data from any database table.

Working With the SmartLink (ISF) Message Cache

Overview of Working with the SmartLink (ISF) Message Cache

Working with the SmartLink (ISF) message cache involves:

- 1 Creating the SmartLink (ISF) message cache database.
- **2** Depending on your connector, inserting one of the following processors:
 - Message Cache Polling Processor
 - Database Polling Processor
- **3** Using the Message Cache Script APIs to do the following:
 - Connect to a message cache
 - Save messages
 - Extract messages

Example of Integrating the SmartLink (ISF) Message Cache

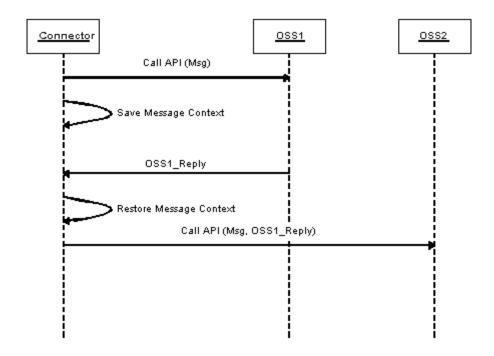
When Synchronizer interacts sequentially and asynchronously with several back ends, an OSS may need to receive messages enriched by one or several previous OSS data. Hence a connector (Synchronizer or OSS Connector) should be able to send messages enriched with custom data collected in past received messages.

For asynchronous exchanges of information between connectors and applications, you may need to save and restore message data between exchanges.

For instance, your application integration might require the following:

- 1 Connector calls an API on OSS 1 but does not wait for the reply
- 2 Connector saves current message context for later use in message cache
- 3 Later connector receives the reply from OSS1 at some point in the future
- 4 Connector restores the message context from message cache

5 Connector calls the OSS 2 API and provides the message context and OSS1 reply



Creating the SmartLink (ISF) Message Cache

Creating the SmartLink (ISF) message cache involves:

• Creating the structure with the messagecacheAdminTool

To create the structure

- 1 Go to <home dir>/bin.
- 2 Run the SmartLink (ISF) Message Cache administration tool. Use the syntax:

```
messagecacheAdminTool create_messagecache_structure
<InstanceName> <ADMIN login> <ADMIN password> [-
cachename:name | -sqlfilename:sql filename] [-quiet]
```

where <db instance>:

Oracle: <instance alias>

- DB2: <database alias>
- SQL Server: <database host> [:<port>] If no port is specified, the tool uses the default SQL server port

When running this tool, you specify at least one of the following:

- -cachename the name of the cache
- -sqlfilename the name of the file containing custom SQL script

When finished, the administration tool displays a confirmation message.

Adding the SmartLink (ISF) Message Cache Processors

Adding the SmartLink (ISF) Message Queue processors to your Integration Process involves:

- Creating processors
- Configuring Message Cache Polling processors
- Configuring Database Polling processors

To create Processors

- 1 Open the Integration Process.
- **2** Right-click *Processors* then select *Create New Processor*. The *Choose a Processor Type* dialog box appears.
- **3** Expand the node of the Processor Type to create.
- **4** Select the Processor to create.
- **5** Enter the name of the Processor.
- **6** Select *Confirm*. The new processor appears in the Connector Explorer and the Integration Process displayed in the Workspace.

To configure Message Cache Polling Processors

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- 3 Select the Message Cache Polling Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
DB User Name	Name to connect to the database
DB User Password	Corresponding password
DB Connection String	Connection string
DB Driver	Database driver
Cache Name	Name of the message cache
Number of threads	Number of threads to allocate
Napping time	Time between polling (in milliseconds)
Max No. Of Messages	Maximum number of messages to extract each time
Custom Extraction SQL	Path to the file containing custom SQL script
User exits class	Class containing user exit code to execute
Unread Messages Only	Boolean. If set to true, extract only unread messages
Save Error Description	Boolean. If set to true, save the description of any error

To configure Database Polling Processors

- 1 Open the Integration Process.
- **2** Expand the *Processors* node.
- **3** Select the Database Polling Processor to modify.
- 4 In the Properties pane, enter the following:

PROPERTY	DESCRIPTION
DB User Name	Name to connect to the database
DB User Password	Corresponding password
DB Connection String	Connection string
Number of threads	Number of threads to allocate
Napping time	Time between polling (in milliseconds)
Flag Read Line	Boolean. If set to true, flags the row as being read
Delete Row On Success	Boolean. If set to true, deletes the row when the extraction occurs without errors
Save Error Description	Boolean. If set to true, saves the description of any error.
User exits class	Class containing user exit code to execute
Identifier Column Name	Name of the column containing the ID name
Identifier Column Type	Type of the column containing the ID
Selection Query	File name of the code used to select messages
Lock Query	File name of the code used to lock messages
Message Factory Class	Class containing the message factory
Confirm Update	SQL file called after successfully processing the message
Confirm Error	SQL file called after processing of a message fails
Confirm Error With Msg	SQL file called after processing of a message fails and the error description field of the message must be updated

Connecting to the SmartLink (ISF) Message Cache

Connecting to the SmartLink (ISF) message cache involves:

- Creating a Script Dispatcher processor
- Inserting the Script Dispatcher in the Integration Process containing the SmartLink (ISF) Message Cache Processors
- Programming the connection script in the Script Dispatcher

To create Processors

- **1** Open the Integration Process.
- **2** Right-click *Processors* then select *Create New Processor*. The *Choose a Processor Type* dialog box appears.
- 3 Expand the node of the Processor Type to create.
- 4 Select the Processor to create.
- **5** Enter the name of the Processor.
- **6** Select *Confirm*. The new processor appears in the Connector Explorer and the Integration Process displayed in the Workspace.

To connect Processors

- 1 Open the Integration Process in the Workspace.
- **2** Find the Processor you want to connect.
- 3 Click the Processor then drag the pointer to the destination Processor. Depending on the type of destination Processor, the following may occur:
 - An arrow appears connecting the two Processors.
 - If more than one type of connection can be made, a menu appears asking you to select the type of connection.
 - A dialog box appears informing you that you cannot connect the Processors. This occurs when:
 - The Processor is already connected
 - The Processors are not compatible

Working with Scripts

1 Open the Integration Process.

- 2 Right-click *Scripts* then select *Create New Script*. The *Create new Script* dialog box appears.
- **3** Enter the name of the script.
- 4 Choose OK. The new script appears.

Before you start working with this script, you must create or specify the script file.

To program the connection script

- **1** Open the script.
- **2** Use the messagecache Script APIs to program you connection.
- 3 Save the script.

For more information about:

- programming scripts, refer to Programming Scripts in this manual
- messagecache Script APIs, refer to the SmartLink (ISF) Script API HTML documentation

To specify the scripts

- **1** Open the *Integration Process*.
- **2** Expand the *Processors* node.
- **3** Select the Script Engine to modify.
- 4 In the Properties pane, select *Add*. A new Script row appears.
- **5** Select the script from the drop-down list.
- 6 Repeat as necessary.

Saving Messages

Saving messages to the SmartLink (ISF) Message Cache involves:

- Connecting to a message cache
- Saving a message
- Extracting a message
- Updating a message

You use the Script APIs to preform these actions.

Example of Saving Messages

The following is an example of implementing the SmartLink (ISF) Message Cache in a simple Integration Process.

If your connector needs to interact asynchronously with an OSS, you can do the following:

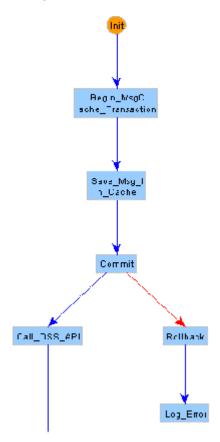
- **1** Parse the inbound message
 - You can use either Xml2hash transmappers or DOM builder processors.
- 2 Save inbound message in message cache
 - You use the Script Dispatcher and program a script using the SmartLink (ISF) Message Cache Script APIs.
- 3 Call OSS system API using inbound message data
 - You use the Script Dispatcher and program a script.
- 4 Update the message with the identifier returned by API

You use the Script Dispatcher and program a script using the SmartLink (ISF) Message Cache Script APIs.

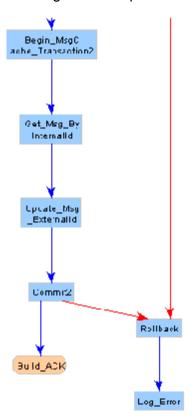
This integration process which saves and updates the message context before calling a OSS API looks like this:



The Call_OSS_API Script Dispatcher processor contains the script which saves the message. The script looks like this:



After calling the OSS API, your Integration Process may need to update the message context with a Reply ID (External Identifier) provided by the OSS. You may need this ID for message. The script looks like this:



Extracting Messages

Saving messages to the SmartLink (ISF) Message Cache involves:

- Connecting to a message cache
- Extracting a message
- Updating a message

You use the Script APIs to preform these actions.

Example of Extracting Messages

The following is an example of implementing the SmartLink (ISF) Message Cache in a simple Integration Process.

If your connector needs to extract messages from the SmartLink (ISF) Message Queue, you can do the following:

1 Connect to a message cache

You can use the Message Cache Poller processor to connect to the cache and extract the messages in the cache inserted by other applications.

2 Build the message

You can use the Script Engine processor to rebuild the message using the information extracted from the cache. The content of the message output is expected to be a DOM object.

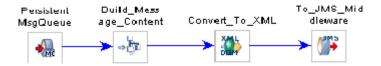
3 Convert the message to XML

You use the DOM2XML processor to convert the DOM message to XML.

4 Send the message

You then send the message to its destination message queue.

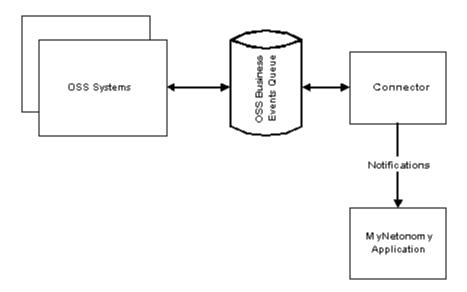
An integration process which extracts and sends the message to an outbound JMS Series Queue looks like this:



Using the SmartLink (ISF) Message Cache to Build a Business Event Queue

You can use the SmartLink (ISF) Message Cache to create an OSS Business Event Queue. An OSS Business Event Queue is a queue you use to collect and store OSS business events.

Your connector can be configured to poll this Business Event Queue at periodic intervals. When any new events are found, your connector can extract them and then trigger specific events. For instance, a business event may trigger sending a notification message to the Synchronizer as shown in this diagram:



New Topic

OSS Business Event Queues must contain the minimum amount of data ot correctly identify the business event. For instance, a simple business event contains:

- Business Event Identifier (mandatory)
- Business Event Type
- Business Event Data

The OSS are responsible for entering the business events in the gueue.

Creating an OSS Business Event Queue

If your connector needs to poll the Business Event Queue, you can do the following:

1 Poll the Business Event Queue

You use a Message Cache Poller processor which is responsible for polling and extracting business events from the cache.

2 Build the message

You can use the Script Engine processor to rebuild the DO message using the information extracted from the cache. The content of the message output is expected to be a DOM object.

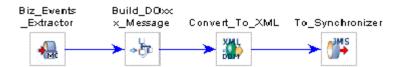
3 Convert the message to XML

You use the DOM2XML processor to convert the DOM message to XML.

4 Send the message

You use the OutboundJMSQueue processor to send the message to the Synchronizer.

This integration process which polls the Business Event Queue and updates your application accordingly looks like this:



Working With the Message Schema Reference

In This Section

About the Message Structure Files	148
About Working with the Message Schema Reference	149
Overview of the Business Message Structure	150
Adding New Messages	153
Overriding Default Outbound Message Mapping	161
Customizing Message Fields	163
Working with Synchronizer Outbound Message Fields	169
Customizing Synchronizer Inbound Messages	170
Customizing Standard Message Structures	171
Generating the Message Schema Reference Files	173

About the Message Structure Files

The Message Schema Reference is a set of XML schema files.

Each message has a XML schema file which specifies:

- The structure of the message
- The mapping of information in the message

By default, these files are located in

<connector_home>/data/schemaref/biz/events

where <connector_home> is the location where the connectors are installed.

These schema files are used to generate the structure XML file and the mapping XML file. These files are used by the connector to read and build messages.

These files adhere to the W3C XML Schema specification. For more information about this specification, refer to the W3C web site at

http://www.w3.org/TR/xmlschema-0/

For the list of messages, refer to Core Messages in this manual.

About Working with the Message Schema Reference

Your application may not require you to modify the structure of format of messages. If you need to, you can customize the structure of messages.

When customizing the Message Schema reference, you do the following:

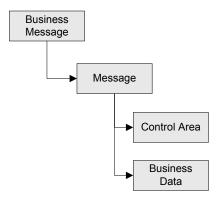
- 1 Create .xsd schema files
- 2 Generate the corresponding XML files using the schemaRefTool

Customizing messages involves:

- Adding New Messages
- Customizing Messages
- Customizing Message Fields
- Customizing Message Structure
- Generating the Message Schema Reference Files

Overview of the Business Message Structure

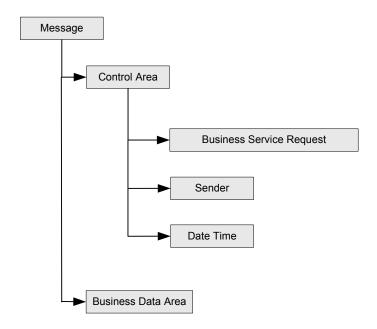
The Message Schema Reference contains the Message Schema structure and mapping definitions. These messages contain technical data and business data organized into different sections:



About the Control Area

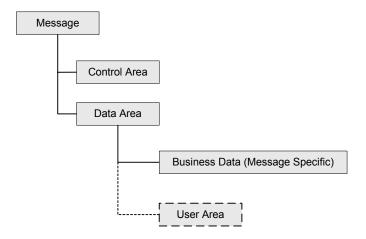
The control area is used to define general message information. This section of the message contains the following:

- The Business Service Request (BSR) qualifies the action that the Sender application wants the Receiver application to perform
- The Sender identifies characteristics and control identifiers that relate to the application which creates the Business message
- The Date Time (creation) segment is used to store the date time information of the message.



About the Business Data Area

The Business Data area contains the data dedicated to the Business message. It also contains the optional customized data the System Integrator adds to adapt it to his specific requirements:



You can extend every core Business Object to add the fields required for any specified implementation, or other business logic object.

Adding New Messages

You can add new messages to the standard messages in order to conform to the integration needs of a project.

This can be useful if you need to send fully customized messages to specific back-end systems that cannot process standard messages.

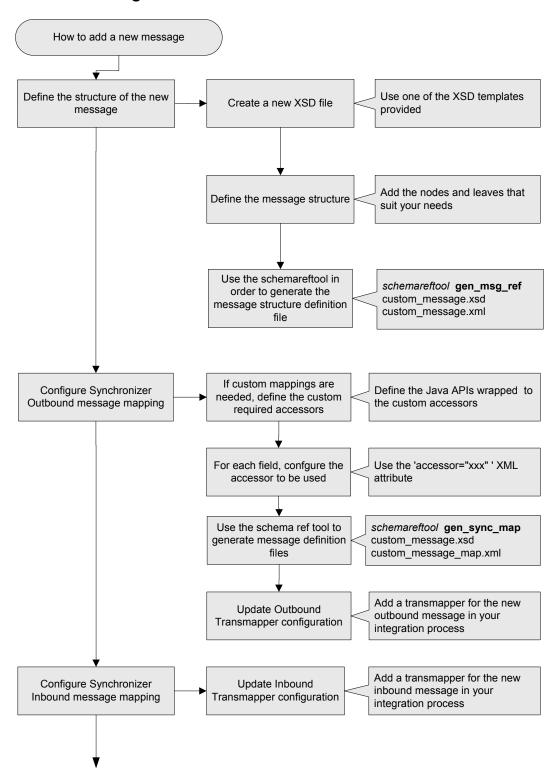
When you exchange messages with OSS Connectors or back end systems, you exchange data with them. These data must be exchanged in a structured manner. The XML format is used to format these data. This means you must define the structure of the XML message in which your are going to put (or get) data.

To create new messages, you can:

- Create a fully customized message in which you define any field you may need with the structure corresponding to you needs.
- Create a message using the standard message structure.

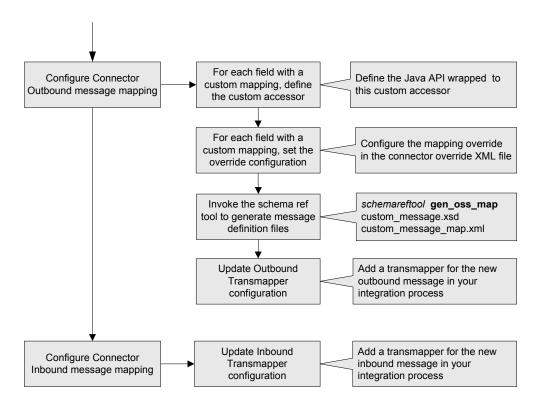
In this case, you use the standard sections (cntrol area, datarea and userarea). Then you customize the content of the datarea and userarea sections to suit your needs.

To add a new message

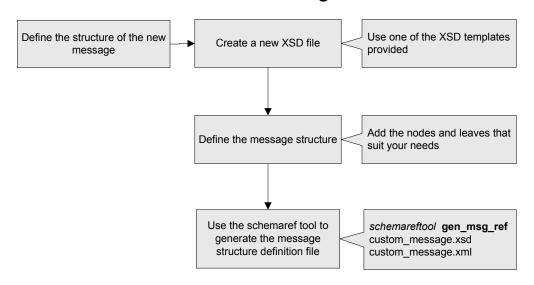


Continued on next page.

To add a new message (Continued)



Define the Structure of the New Message



To define the structure of the new message:

- 1 Create a new XSD file located in the custom events directory (<home dir>/data/schemaref/biz/custom).
 - If you want to create a new, fully customized message, use the NewMessage Template.xsd template.
 - If you do not want to create a new message, use the NewMessageStd Template.xsd template.

Both templates are provided in Documentation/samples/Schemaref templates.

2 In the XSD file, define your message structure: add the XML nodes and leaves that suit your needs.

You must name the root tag of the message "request"

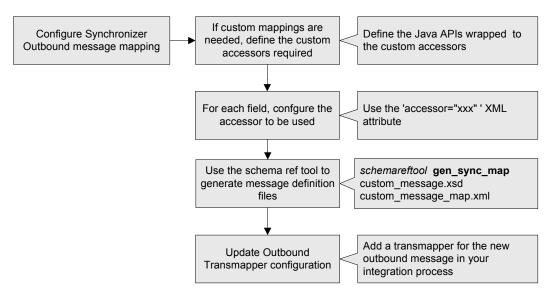
You must name the first child tag of this root tag as the message type (for example REQADDSERVICE).

- 3 Once you have defined your message, generate the message structure definition files that will be used by the Synchronizer or Connector Template. Use the "gen_msg_ref" command of the schemareftool. You must provide the schemareftool with the following parameters:
 - custom message definition (full path to your XSD file)
 - destination message structure definition file (XML file). This file should be located in <home dir>/share/schemaref

Example:

```
CUST_EVENTS=/<home_dir>/data/schemaref/biz/custom
SHARE_REF=/<home_dir>/share/schemaref
/<home_dir>/bin/schemareftool_gen_msg_ref
$CUST_EVENTS/reqaddservice_custom.xsd
$SHARE_REF/reqaddservice.xml
```

Configure Synchronizer Outbound Message Mapping



- 1 If you need to use custom java APIs to fill the content of your fields, you must define custom accessors. For each custom accessor used, you must declare the custom java API that will be wrapped to it in the Macro file accessors custom.properties configuration file.
- 2 Once you have defined the accessors (either as custom or core), you must use the 'accessor' XML attribute to configure which accessor must be called for each field. You must create this accessor configuration in a dedicated sub-section called "appinfo" within an "annotation" section of the field you are defining:
 - You must use the following template:

For each outbound message field, you can choose one of the following options:

- generate the field
- do not generate the field
- generate the field but with no content (empty field). This option is only applicable for leaves.

The default action is to generate the field. If you do not define any specific configuration, your new fields will be automatically present in your outbound message. For more information, refer to To activate How to activate/deactivate a field."

- 3 Once you have defined the field mapping configuration, you must generate the message mapping definition files for the Synchronizer. Use the "gen_sync_map" command of the schemareftool. You must provide the schemaref tool with the following parameters:
 - (mandatory) custom message definition (full path to your XSD file)
 - (mandatory) destination message mapping definition file (XML file). This file should be located (or created) in

```
<home dir>/config/synchronizer/transmapper/accessor
```

 (optional) message field activation configuration file (XML file). This file should be located (or created) in

```
<home dir>/data/schemaref/biz/events/activation
```

Example:

```
CUST_EVENTS=/<home_dir>/data/schemaref/biz/custom

SYNC_MAP=/<home_dir>/config/synchronizer/transmapper/access
or

/<home_dir>/bin/schemareftool gen_sync_map
$CUST_EVENTS/reqaddservice_custom.xsd
$SYNC_MAP/reqaddservice map.xml
```

Configure Synchronizer Inbound Message Mapping

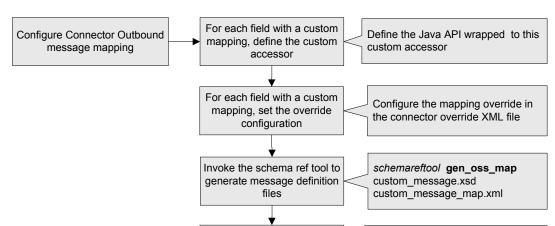


Every field of an inbound message is automatically mapped by the Synchronizer inbound transmapper default configuration. You do not have to configure any specific message field mapping. The default behavior is to map leaves to java strings and nodes to hashtables.

Add a transmapper for the new

outbound message in your

integration process



Configure Connector Outbound Message Mapping

1 If you need to use custom java APIs to fill the content of your fields, you must define custom accessors. For each custom accessor used, you must declare the custom java API that will be wrapped to it in the Macro file accessors custom.properties configuration file.

Update Outbound

Transmapper configuration

- 2 The Connector outbound mapping configuration for a specific message is automatically generated based on that message's structure definition file. If you want to customize a message field mapping, you must override the default mapping configuration of this field.
 - You can override every message field mapping configuration by specifying the field name (and path) and the new accessor to be used in the XML override configuration file of your connector. For more information, refer to *To override a connector default mapping configuration*.
- 3 Once you have configured the field mapping override, you must generate the message mapping definition files for your connector. Use the gen_oss_map command of the schemareftool. You must provide the schemaref tool with the following parameters:
 - (mandatory) custom message definition (full path to your XSD file)
 - (mandatory) destination message mapping definition file (XML file). This file should be located in
 - <home dir>/config/yourconnector/transmapper/accessor
 - (optional) core mapping override configuration file (XML file).
 - (mandatory if core override files specified and xsd file not in custom directory)
 core mapping directory (<home dir>/data/connectors/)

Example:

```
CUST_EVENTS=/<home_dir>/data/schemaref/BIZ/custom

OVERRIDE_DIR=/<home_dir>/data/connectors

LPBK_MAP=/<home_dir>/config/loopback/transmapper/accessor
/<home_dir>/bin/schemareftool_gen_oss_map
$CUST_EVENTS/reqaddservice_custom.xsd
$LPBK_MAP$/reqaddservice_map.xml -core_override
connectortemplate.xml -core_override dir $OVERRIDE_DIR
```

Configure Connector Inbound Message Mapping



Every field of an inbound message is automatically mapped by your Connector inbound transmapper default configuration. You do not have to configure any specific message field mapping. The default behavior is to map leaves to java strings and nodes to hashtables.

Overriding Default Outbound Message Mapping

The Connector outbound mapping configuration for a specific message is automatically generated based on that message's structure definition file. As a consequence, if you want to customize a message field mapping, you must override the default mapping configuration of this field.

You can override a message field mapping configuration by specifying the field name (and path) and the new accessor to be used in the XML override configuration file of your connector.

To override default outbound message mapping

- 1 Create an XML file using the template provided in Documentation/Samples/Schemaref templates/OverrideConfig Template.xml
- 2 For each field for which you want to change the mapping configuration, you must create one new element which specifies which accessor to use. The elements must be grouped by message type and then by element type. The following XML code gives you a sample configuration:

In order to apply your changes, you must use the schemarefTool with the gen_oss_map command to generate your Connector outbound message mapping configuration. Specify the XML file you created in step one as the last parameter (- custom_override) of the gen_oss_map command.

Note: If you do not use the <code>core_override</code>, your custom override has to define the default mapping for node and leaf elements:

```
<outbound_mapping_override
custom_node_accessor="getopt($ELEMENTNAME)"
custom_leaf_accessor="getopt($ELEMENTNAME)"
use_custom_accessors="true">)
```

Customizing Message Fields

Customizing core messages involves:

- Adding or extending message fields
- Activating or deactivating message fields

Restrictions on customizing core messages:

- Synchronizer restrictions
 - New fields must be appended to existing fields
 - Field order cannot be changed
- Connector restrictions
 - Message structure definitions can only be extended. They cannot be updated.
 - Message mapping definitions can be customized (overload mechanism).

The available core messages are listed in *Core Messages*.

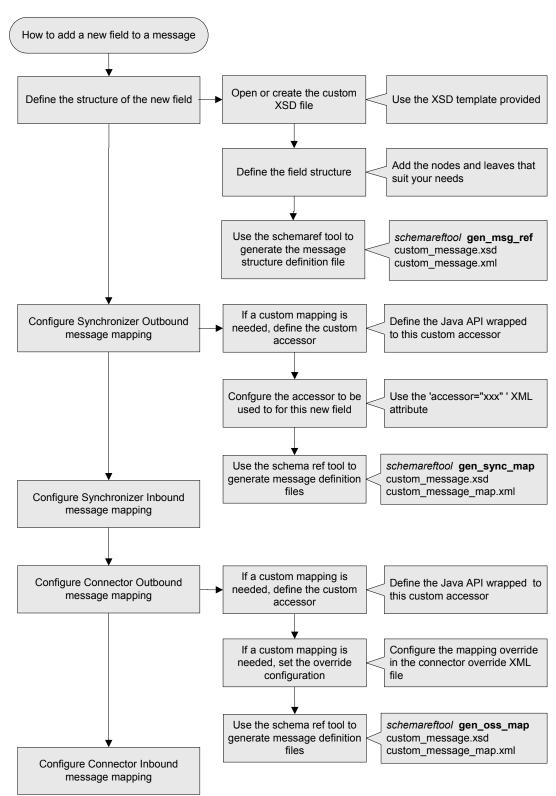
You might add a message field if you need to send additional data to the back-end systems contained in specific tags. You can also carry out specific processing tasks (such as message routing etc.) by using user-defined fields in the Synchronizer's inbound messages.

You can customize message content in two different ways.

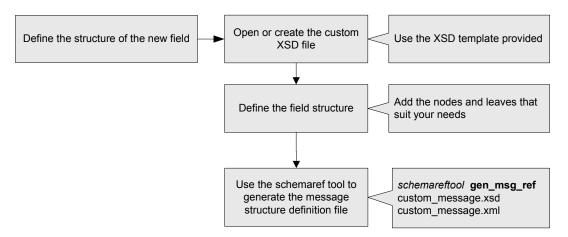
You can add any desired field to the custom section of standard fields.

You can also add any desired field in the Userarea of each message

To add a new message field



Define the Structure of the New Field



To define the structure of the new field:

1 If you have not yet customized any fields in your message, you must create a new XSD file create it in the custom events directory

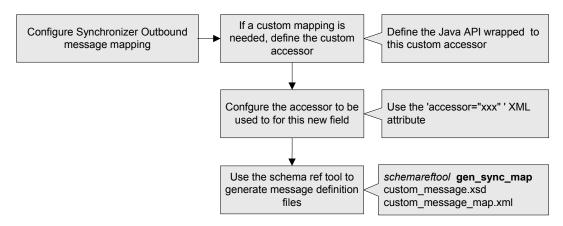
(<home_dir>/data/schemaref/biz/custom). Use the
CustomMessage_Template.xsd template provided in
Documentation/samples/Schemaref templates. To customize a core
message you must update the reference (<xs:redefine
schemaLocation="../events/reqaddservice.xsd") to the core message
you want to extend.</pre>

- 2 Once you have created your custom XSD file, open it and define your field structure: add the XML nodes and leaves that suit your needs.
- 3 Once you have defined your field, you must generate the message structure definition files that will be used by the Synchronizer or Connector Template. Use the gen_msg_ref command of the schemareftool. You must provide the schemaref tool with the following parameters:
 - (mandatory) custom message definition (full path to your custom XSD file)
 - (mandatory) destination message structure definition file (XML file). Create it in <home dir>/share/schemaref

Example:

```
CUST_EVENTS=/<home_dir>/data/schemaref/biz/custom
SHARE_REF=/<home_dir>/share/schemadef
/<home_dir>/bin/schemareftool_gen_msg_ref
$CUST_EVENTS/reqaddservice_custom.xsd
$SHARE_REF/reqaddservice.xml
```

Configure Synchronizer Outbound Message Mapping



- 1 If you need to use custom java APIs to fill the content of your fields, you must define custom accessors. For each custom accessor used, you must declare the custom java API that will be wrapped to it in the Macro file accessors custom.properties configuration file.
- 2 Once you have defined the accessors (either as custom or core), you must configure which accessor must be called for your new fields using the 'accessor' XML attribute. You must create this accessor configuration in a dedicated subsection called "appinfo" within an "annotation" section of the field you are defining.

You must use the following template:

For each outbound message field, you can choose one of the following options:

- generate the field
- do not generate the field
- generate the field but with no content (empty field). This option is only applicable for leaves.

The default action is to generate the field. If you do not define any specific configuration, your new fields will be automatically present in your outbound message. For more information, refer to "How to activate/deactivate a field."

- Once you have defined the filed mapping configuration, you must generate the message mapping definition files for the Synchronizer. Use the "gen_sync_map" command of the schemareftool. You must provide the schemaref tool with the following parameters:
 - (mandatory) custom message definition (full path to your XSD file)

- (mandatory) destination message mapping definition file (XML file). This file should be located (or created) in
 - <home_dir>/config/synchronizer/schema_map/accessor
- (mandatory if custom message is not located in the custom folder) location of the core messages (<home_dir>/data/schemaref/biz/events)
- (optional) message fields activation configuration file (XML file). This file should be located (or created) in

```
<home dir>/data/schemaref/biz/events/activation
```

Example:

```
CUST_EVENTS=/<home_dir>/data/schemaref/biz/custom

SYNC_MAP=/<home_dir>/config/synchronizer/schema_map/acces sor

/<home_dir>/bin/schemareftool gen_sync_map

$CUST_EVENTS/reqaddservice_custom.xsd

$SYNC_MAP/reqaddservice_map.xml -schemaref_dir

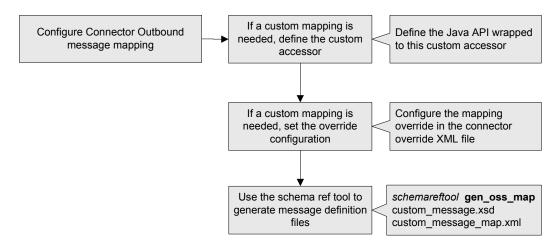
<home_dir>/data/schemaref/biz/events
```

Configure Synchronizer Inbound Message Mapping

Configure Synchronizer Inbound message mapping

Every field of an inbound message is automatically mapped by the Synchronizer inbound transmapper default configuration. You do not have to perform any specific message field mapping configuration. The default behavior is to map leaves to java strings and nodes to hashtables.

Configure Connector Outbound Message Mapping



- 1 If you need to use custom java APIs to fill the content of your fields, you must define custom accessors. For each custom accessor used, you must declare the custom java API that will be wrapped to it in the Macro file accessors_custom.properties configuration file.
- 2 The Connector outbound mapping configuration for a specific message is automatically generated based on that message's structure definition file. If you want to customize a message field mapping, you must override the default mapping configuration of this field.
 - You can override every message field mapping configuration by specifying the field name (and path) and the new accessor to be used in the XML override configuration file of your connector. For more information, refer to *To override a connector default mapping configuration*
- 3 Once you have configured the field mapping override, you must generate the message mapping definition files for your connector. Use the <code>gen_oss_map</code> command of the schemareftool. You must provide the schemaref tool with the following parameters:
 - (mandatory) custom message definition (full path to your XSD file)
 - (mandatory) destination message mapping definition file (XML file). This file should be located in

```
<home dir>/config/yourconnector/schema map/accessor
```

 (optional and multiple) mapping override configuration files (XML file). These files should be located in <home dir>/data/connectors/

Example:

```
CUST_EVENTS=/<home_dir>/data/schemaref/biz/custom

LPBK_MAP=/<home_dir>/config/loopback/schema_map/accessor
/<home_dir>/bin/schemareftool gen_oss_map
$CUST_EVENTS/reqaddservice_custom.xsd
$LPBK_MAP/reqaddservice_map.xml
```

Configure Connector Inbound Message Mapping

Configure Connector Inbound message mapping

Every field of an inbound message will be automatically mapped by your Connector inbound transmapper default configuration. You do not have to configure any specific message field mapping. The default behavior is to map leaves to java strings and nodes to hashtables.

Working with Synchronizer Outbound Message Fields

You can activate or deactivate every field of the standard messages generated by the Synchronizer in order to suit your back end systems requirements.

When the standard messages do not contain enough information to suit your needs, or, on the contrary you want to reduce message contents, you might need to activate or deactivate certain message fields.

For each field of a message sent by the Synchronizer you can choose one of the following options:

- Activate the message field: The field will be mapped with its configured accessor and may be present in the Synchronizer outbound message depending on the accessor context.
- Deactivate the message field: In this case you have two options:
 - do not to generate the message field
 - generate an empty XML field (<yourfieldname/>).

To deactivate or activate a field

- 1 Go to data/schemaref/biz/events/activation.
- **2** Open the XML file corresponding to your message.
- **3** Locate the field you want to deactivate in the XML file.
- 4 If you want to deactivate a message field, that is to say to generate an empty XML field, add the following attribute:

```
<yourfieldname tag="empty"/>
```

5 If you want to to deactivate a message field but **not** to generate the XML field, add the following XML attribute:

```
<yourfieldname tag="null"/>
```

To Activate a field:

If you want to activate a field, you have two options. Either:

- Do not specify any "tag" attribute, or
- Do not add your field in the activation file
- 6 In order to apply your changes, you must use the schemarefTool with the gen_sync_map command to generate your Synchronizer outbound message mapping configuration. Specify the XML file you created in step one as the last parameter (-activation file) of the gen sync map command.

Customizing Synchronizer Inbound Messages

You can customize a Synchronizer inbound message by adding fields (leaves or nodes). These new fields are automatically mapped as strings in the message hash table.

If you have defined additional sections (for example in the userarea) in the Synchronizer inbound messages, you might need to check some fields' values within these sections in order to perform specific message routing.

You can retrieve and use your custom fields' values through the workflow APIs, for example: "Message.getElement".

To use additional fields in Synchronizer inbound messages

- 1 Create a new script engine.
- 2 Insert the component in the inbound XML parser after the first Transmapper.
- 3 Create a new Script.
- 4 In the Script, use the API Message.getElement() to retrieve your value
- **5** Use the value retrieved to (for example):
 - route the message to another SmartLink (ISF) component
 - call an external API

Customizing Standard Message Structures

The Synchronizer uses messages to exchange data. These messages use the XML format, from the XML W3C specification, and consist of nodes and leaves. If necessary, you can add leaves or nodes to a message and add their respective mappings in order to fill them.

The nodes and leaves from the standard Telco Service & Analytics Manager messages are organized as follows:

- Root element (<request>)
 - Control area (<cntrolarea>)
 - Business Service Request area (<BSR>)
 - Sender area (<sender>)
 - Date and time area (<datetime>)
 - Data area
 - User Area

Only the Data area and the User area can be customized as follows:

For the Data area, only XML nodes that contain a dedicated "custom" tag can be modified. Within these "custom" tags, you can add any node or leaf you may need.

The additional parameter description is also put in this dedicated section.

For the User area, you can add any node or leaf you may need, this area is dedicated to your requirements.

You may need to add specific nodes or leaves in the User area. For example when you have to provide back-end systems with specific routing information, such as a specific system identifier.

If you have added additional parameters to standard Telco Service & Analytics Manager objects, the structure of these additional parameters is already defined. You do not have to define the structure, either for outbound messages or for inbound messages.

To add a custom field

- 1 Open the message structure description in the message schema reference
- 2 Add your nodes or leaves in the custom sections of objects as necessary:
 - in the data area
 - in the user area

You can use the standard include mechanisms of the W3C XML Schema.

- 3 Save your description in a new file
 - We recommend you **do not** to replace the standard message definition
- **4** Run the convert tool by specifying your previously created file as the input file for the tool

Generating the Message Schema Reference Files

You use the schemarefTool to generate the XML message definition files from the Message Schema reference.

You use this tool to generate the following from the Message Schema Reference:

- Message structure definition files
- Message documentation files
- Message mapping definition files for Synchronizer connectors
- Message mapping definition files for OSS Connectors

To generate message structure definition files

- 1 Go to <ils dir>/bin.
- **2** Run the schemarefTool tool. Use the syntax:

```
schemarefTool gen_msg_ref <xsd_source_file>
<xsd_destination_file> [-quiet]
.
```

where:

<xsd_source_file> is the XML schema file

<xsd_destination_file> is the destination XML structure definition file

To generate message documentation files

- 1 Go to <ils dir>/bin.
- **2** Run the schemarefTool tool. Use the syntax:

```
schemarefTool gen_msg_doc <xsd_src_file> <xsd_dest_struct> [-
activation_file <file>] [-quiet]
```

where:

<src_file> is the XML schema file

<dest_struct> is the destination XML structure definition file (without includes)

-activation file<file> is the file containing activation configuration

To generate message mapping definition files for Synchronizer connectors

1 Go to <ils_dir>/bin.

2 Run the schemarefTool tool. Use the syntax:

```
schemarefTool gen_sync_map <xsd_src_file> <mapping_dest_file>
[-schemaref_dir<dir>] [-activation_file <file>] [-quiet]
where:
<xsd_src_file> is the XML schema file
<mapping_dest_file> is the destination XML mapping definition file
-schemaref_dir<dir> is the directory containing core message definitions
-activation file<file> is the file containing activation configuration
```

To generate message mapping definition files for OSS Connectors

- 1 Go to <ils_dir>/bin.
- 2 Run the schemarefTool tool. Use the syntax:

```
schemarefTool gen_oss_map <xsd_source_file>
<mapping_destination_file> [-core_override <file> ] [-
core_override_dir <dir>] [ -custom_override <file<,file>> ]
[-quiet]
```

where:

<xsd source file> is the XML schema file

-core override<file> is the xml file defining default mapping

-core_override_dir<dir> is the directory where the core_override files are located

-custom_override <file<, file>> is a comma separated list of custom
mappings

CHAPTER 9

Generating Your Connector

In This Section

About Generating Your Connector	176
To generate connectors	
To generate connectors using the Connector Generator	

About Generating Your Connector

The ILS uses the Integration Logic Connector Design file (<connector_name>.ilcd) to store the design and configuration of your connector. This file is for configuring and designing connectors. In order to run your connector, you need to generate the runtime files.

The generated connector runtime files are:

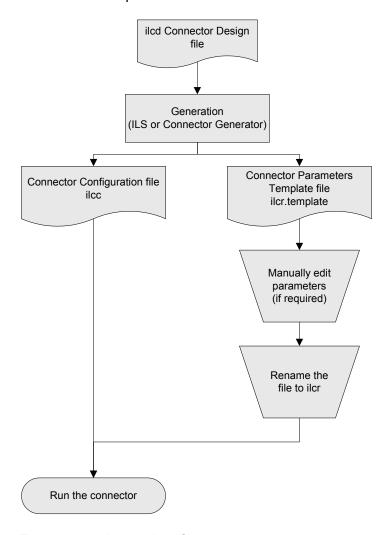
Integration Logic Connector Configuration file (<connector_name>.ilcc)
 This file is an internal configuration file used by the connector. It contains the configuration of your connector.

This file is automatically generated and should not be edited. Use the ILS to change your configuration.

• Integration Logic Connector run time configuration file template (<connector name>.ilcr.template)

This run time configuration file contains the parameters defined for your connector. You edit this file manually and change its name.

The Generation process is as follows:



To generate the runtime files, you can:

- Use the ILS
- Use the Connector Generator command line tool

To generate connectors

- **1** Choose *Connector* > *Generate*. The Generate Connector Configuration dialog box appears.
- 2 If you want the ILS to validate the connector before generation, select *Perform Validation before Generation*.

- 3 In Output directory, enter the location of the generated connector files.
- **4** Choose *Generate*. The ILS generates the following connector runtime files:
 - <connector_name>.ilcc
 - <connector name>.ilcr.template
- **5** When finished, the ILS displays a message in the message pane.
- 6 If you need to change the runtime parameters before running your connector, open <connector name>.ilcr.template and edit the parameters.
- 7 Change the file name to <connector name>.ilcr.

To generate connectors using the Connector Generator

- 1 Go to <ils dir>/bin.
- 2 Run the ilccGenerator tool. Use the syntax:

```
ilccGenerator <connector_name>
```

where <connector_name> is the full path of the Integration Logic Connector Design .ilcd file

The tool generates the the following runtime files in the same directory as the Integration Logic Connector Design .ilcd file.

- <connector name>.ilcc
- <connector name>.ilcr.template
- 3 If you need to change the runtime parameters before running your connector, open <connector_name>.ilcr.template and edit the parameters.
- 4 Change the file name to <connector name>.ilcr.

CHAPTER 10

Running Your Connector

In This Section

About Running the Connector	180	
Specifying Processing Priority	181	
Starting and Stopping the Synchronizer Connector	183	
Starting and Stopping the OSS Connector	184	
Administrating the Connector	185	

About Running the Connector

Running your Connector involves:

- Specifying the processing priority of messages
- Starting and stopping the Synchronizer connector
- Starting and stopping the OSS connector
- Administrating connectors

Specifying Processing Priority

You can assign the priority of messages the Connector handles.

The role of the user generating the request in the CID can be configured to determine the priority of processing by the Connector. For instance, your connector can be configured to process requests created by Dealers before handling requests made by guest users.

For more information about users and roles, refer to *Managing Access to BLM Objects* in the *BLM Reference Guide*.

To assign a priority to a role

In the ACTION_PRIORITY table, add a new record then enter the following mandatory information:

- ACTION CODE This number is the action. This code comes from the ACTION table.
- ROLE ID The number of the role to assign the priority.
- PRIORITY_ID The number of the priority to assign. This ID comes from the PRIORITIES table.

To configure the priority

In the PRIORITIES table, change the values in the PRIORITY_DISTRIB column. The value in this column corresponds to a percentage of messages processed.

The default priorities:

NAME	PRIORITY_ID	PRIORITY_DISTRIB
High	1	50
Medium	2	30
Low	3	20

The total of the values in this PRIORITY_DISTRIB column must add up to 100.

Example of Setting Priorities

For example, the request queue has the following number of requests (with its corresponding $PRIORITY_ID$):

- 55 Create Organization requests (PRIORITY_ID = 1)
- 33 Create Contract requests (PRIORITY ID = 2)
- 22 Add Service requests (PRIORITY_ID = 3)

If you use the default settings, the Connector processes the requests in the following order:

- **5** 50 Create Organization requests
- 6 30 Create Contract requests
- 7 20 Add Service requests
- 8 5 Create Organization requests
- **9** 3 Create Contract requests
- 2 Add Service requests

Starting and Stopping the Synchronizer Connector

You use a set of administration tools to start and stop the Synchronizer connector. The administration tools are:

- agentstart
- agentadm

These administration tools are located in <home dir>/bin.

About Synchronizer Run Modes

The Synchronizer connector works in one of the following modes:

MODE	DESCRIPTION
NORMAL	THE DEFAULT MODE IN WHICH THE SYNCHRONIZER SCANS THE CID AT A REGULAR, FIXED TIME INTERVALS AND EXTRACTS THE REQUESTS
PAUSED	THE SYNCHRONIZER CONNECTOR STOPS SCANNING THE CID FOR REQUESTS, AND WAITS FOR A RESTART COMMAND BEFORE IT RESUMES
RECOVER	THE SYNCHRONIZER CONNECTOR MAKES A FINAL ATTEMPT TO SEND REQUESTS (WITH THE STATUS 'TRANSPORTFAILED') THAT HAVE NOT BEEN SENT DUE TO A TRANSPORT PROBLEM. WHEN FINISHED, IT GOES BACK TO THE NORMAL MODE.

To start the Synchronizer connector

- 1 Go to <home dir>/bin.
- **2** Run agentstart. Use the syntax:

```
agentstart <connector_name>
```

The connector is loaded and it starts the processes it needs.

To stop the Synchronizer connector

- 1 Go to <home dir>/bin.
- 2 Run agentadm. Use the syntax:

```
agentadm <host> <port> shutdown
```

When finished, it displays a message.

Starting and Stopping the OSS Connector

You use a set of administration tools to start and stop the OSS connector. The administration tools are:

- ossstart
- ossadm

These administration tools are located in <home_dir>/bin.

To start the OSS connector

- 1 Go to <home_dir>/bin.
- **2** Run ossstart. Use the syntax:

```
ossstart <connector name>
```

The connector is loaded and it starts the processes it needs.

To stop the OSS connector

- **1** Go to <home_dir>/bin.
- 2 Run ossadm. Use the syntax:

```
ossadm <host> <port> <connector> shutdown
```

When finished, it displays a message.

Administrating the Connector

You can stop, start and manage the connector interactively.

You use a set of administration tools to start, stop and manage the Synchronizer and OSS Connector.

About Administrating Connectors

To administer the connectors, you have a complete set of commands you can use to start, stop, or change the settings of the connector while it runs.

Whenever you send a command, the connector responds confirming the execution of your command. For example, you may see:

```
AGENTADM LOCALHOST 3000 STOP
-200AGENT PAUSED
AGENTADM LOCALHOST 3000 START
-200AGENT RESUMING TO NORMAL MODE
AGENTADM LOCALHOST 3000 POSE
-220UNKNOWN COMMAND POSE
```

To manage the Synchronizer connector

- **1** Log into the computer
- **2** Do one of the following:
 - If the connector is running, run agentadm
 - If the connector is not running, run agentstart <synchronizer_name>, then agentadm
- **3** Use one of the commands below to manage the Synchronizer connector:

COMMAND	DESCRIPTION	PARAMETERS	RETURN VALUES
start	Resumes the execution of the connector. Use this command to restart the connector	none	-200 AGENT START
stop	after a stop command. Stops the connector.	none	-120 AGENT STOP
Stop	The connector no longer processes inbound and outbound messages	Hone	120 NOLINI GTOI
shutdown	Terminates the connector	none	-100 AGENT SHUTDOWN
setparameter	Changes the settings of the remotely.	AGENTNAPPING AGENTLOOPING ONE_EXECUTION	-510 AGENT SET <parameter name="">=<parameter value=""></parameter></parameter>
	Use the following syntax:	NBRROW	
	setparameter <parameter>=<value></value></parameter>	QFILL	
		MAXNUMBERRETRY RETRYDELAY	
getparameter	Restarts the current value of the parameter	AGENTNAPPING AGENTLOOPING ONE_EXECUTION	-610 AGENT GET <parameter name="">=<parameter value=""></parameter></parameter>
		NBRROW QFILL MAXNUMBERRETRY RETRYDELAY	
		Any parameter defined in the properties files	
getmode	Displays the current running mode	none	-210 AGENT NORMAL
			-220 AGENT REDUCE
			-230 AGENT PAUSE
getstatus	Displays the current state	none	-130 AGENT NORMAL
	otato		-140 AGENT REDUCE
			-150 AGENT PAUSE

force	Forces the connector to run in the specified mode	NORMAL REDUCE RECOVER	-410 AGENT SET MODE=NORMAL -410 AGENT SET MODE=REDUCE -410 AGENT SET MODE=RECOVER"
			For MODE=RECOVER, an additional parameter must be provided: FILTER <filtername></filtername>
info	Displays information about the connector	none	[AGENTNAME:xxx,NBTHREADS:xxx,FILL:xxx,NBROW:xxx,NAPPING:xxx,LOOPING:xxx,VERSION:xxx,BUILD:xxx,OS:xxx]"
kill	Forces shutdown	none	none
list	Returns all the profiling information	none	none
purge	Purges the profiling information	none	none
stat	Returns the profiling information of the last profiling element	none	none
version	Returns the version of the connector	none	none

To manage the OSS connector

- 1 Log into the computer
- 2 Do one of the following:
 - If the connector is running, run ossadm
 - If the connector is not running, run ossstart <synchronizer_name>, then ossadm
- **3** Use one of the commands below to manage the OSS connector:

COMMAND	DESCRIPTION	PARAMETERS	RETURN VALUES
START	Resumes the execution of the connector.	none	-200 AGENT START
	Use this command to restart the connector after a stop command.		
stop	Stops the connector.	none	-120 AGENT STOP
	The connector no longer processes inbound and outbound messages		
shutdown	Terminates the connector.	none	-100 AGENT SHUTDOWN
kill	Forces shutdown.	none	none

APPENDIX A

SmartLink (ISF) Tool Reference

In This Section

About the SmartLink (ISF) Administration Tools	190
Synchronizer Connector Administration Tools	191
OSS Connector Administration Tools	195
SmartLink (ISF) Administration Tool	196
Message Schema Reference Tool	197
	201

About the SmartLink (ISF) Administration Tools

A complete set of administration tools helps you manage the different components of your TSM.

The name of the files depends on your operating system. For Windows, the tools have a .cmd extension and for UNIX a .sh extension. The behavior of these tools is identical and they have the same command line options.

These administration tools are located in <home dir>/bin.

Make sure that any required files are in the CLASSPATH and LIB_PATH specified in the pre.env environment setting file. This file is located in <home dir>/bin/env.

Synchronizer Connector Administration Tools

You use the following administration tools to administrate the Synchronizer Connector:

- agentstart
- agentadm

These tools are located in <home_dir>/bin.

agentstart Syntax

agentstart <connector name>

PARAMETERS	DESCRIPTION
<connector_name></connector_name>	Name of the synchronizer connector to start

agentadm Syntax

agentadm help | <host> <port> <command> [<parameters>]

PARAMETERS	DESCRIPTION
help	Displays help for the tool
<host></host>	Specifies the agent host
<port></port>	Specifies the agent administration port
<command/>	Administration command

COMMAND	DESCRIPTION	PARAMETERS	RETURN VALUES
start	Resumes the execution of the connector.	none	-200 AGENT START
	Use this command to restart the connector after a stop command.		
stop	Stops the connector.	none	-120 AGENT STOP
	The connector no longer processes inbound and outbound messages		
shutdown	Terminates the connector	none	-100 AGENT SHUTDOWN
setparameter	Changes the settings of the connector remotely.	AGENTNAPPING AGENTLOOPING ONE_EXECUTION	-510 AGENT SET <parameter name>=<parameter value=""></parameter></parameter
	Use the following syntax:	NBRROW	
	setparameter <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>	QFILL	
		MAXNUMBERRETRY RETRYDELAY	

	I		
getparameter	Restarts the current value of the parameter	AGENTNAPPING AGENTLOOPING ONE_EXECUTION NBRROW QFILL MAXNUMBERRETRY RETRYDELAY Any parameter defined in the properties files	-610 AGENT GET <parameter name>=<parameter value=""></parameter></parameter
getmode	Displays the current running mode	none	-210 AGENT NORMAL -220 AGENT REDUCE -230 AGENT PAUSE
getstatus	Displays the current status	none	130 AGENT NORMAL -140 AGENT REDUCE -150 AGENT PAUSE
force	Forces the connector to run in the specified mode	NORMAL REDUCE RECOVER	-410 AGENT SET MODE=NORMAL -410 AGENT SET MODE=REDUCE -410 AGENT SET MODE=RECOVER For MODE=RECOVER, an additional parameter must be provided: FILTER <filtername></filtername>
info	Displays information about the connector	none	[AGENTNAME:xxx,NBTHRE ADS:xxx,FILL:xxx,NBROW:x xx,NAPPING:xxx,LOOPING: xxx,VERSION:xxx,BUILD:xx x,OS:xxx]
kill	Forces shutdown	none	none
list	Returns all the profiling information	none	none
purge	Purges the profiling information	none	none

stat	Returns the profiling information of the last profiling element	none	For each message type, displays the following information: - name of message type - min processing time - max processing time - total processing time - deviation processing time - nb number of messages - probability the percentage of successful messages processed - first timestamp of the first message processed - last timestamp of the last message processed - throughput number of messages processed
version	Returns the version of the connector	none	none

OSS Connector Administration Tools

The tools you use to administrate the OSS connector are:

- ossstart
- ossadm

ossstart Syntax

ossstart help | <connector name>

PARAMETERS	DESCRIPTION
<connector_name></connector_name>	NAME OF THE OSS CONNECTOR TO START

ossadm Syntax

ossadm help | <host> <port> <command>

PARAMETERS	DESCRIPTION	
HELP	DISPLAYS HELP FOR THE TOOL	
<host></host>	Specifies the connector host	
<port></port>	Specifies the connector port	
<command/>	Administration command	

COMMAND	DESCRIPTION	PARAMETERS	RETURN VALUES
SHUTDOWN	TERMINATES THE CONNECTOR	NONE	NONE
KILL	FORCES SHUTDOWN	NONE	NONE

SmartLink (ISF) Administration Tool

You use the isfadm tool to manage the SmartLink (ISF).

isfadm Syntax

isfadm help | <host> <port> <objecttype> <object> <command>

PARAMETERS	DESCRIPTION	
HELP	DISPLAYS HELP FOR THE TOOL	
<host></host>	SPECIFIES THE HOST TO USE.	
<port></port>	SPECIFIES THE ADMINISTRATION PORT TO USE	
<objectype></objectype>	The SmartLink (ISF) component to manage:	
	connector	
	• queue	
	• var	
• <object></object>	The object of the command.	
• <command/>	Administration command	

COMMAND	DESCRIPTION	PARAMETERS	RETURN VALUES
START	RESUMES THE EXECUTION OF THE COMPONENT.	NONE	
STOP	STOPS THE COMPONENT.	NONE	
GETSTATUS	DISPLAYS THE CURRENT STATUS	NONE	

Message Schema Reference Tool

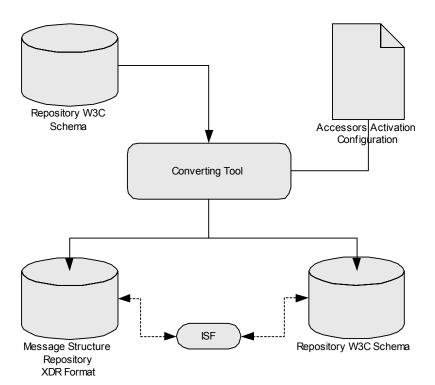
This tool handles the Message Schema Reference. This tool handles the following commands:

- Gen msg ref: Generates the message structure definition files
- Gen msg doc: Generates the message documentation files
- Gen_synch_map: Generates the message mapping definition files for the Synchronizer
- Gen_oss_map: Generates the message mapping definition files for the Connectors

You use the converting tool to:

- Transform message structure and/or mapping definitions from the W3C format into the XDR format.
- Apply the accessors' activation configuration (specified by the user).

If no configuration is available, all the accessors are generated.



schemarefTool Syntax

schemarefTool help | <command> [<parameters>] <help>

PARAMETERS	DESCRIPTION
help	Displays help for the tool
<command/>	Message Schema Reference Tool command
<pre><parameters></parameters></pre>	Specifies the parameters for the command
<command/> help	Displays help for the command

COMMAND	DESCRIPTION	PARAMETERS
gen_msg_ref	generates the message structure definition files	<xsd_source_file>: Source file containing the XML Schema definition of the message</xsd_source_file>
		<pre><xsd_destination_file>: Destination file in which the structure definition in old XML format will be saved</xsd_destination_file></pre>
		-acitivation_file <file> (Optional) File containing the activation configuration for the file being processed</file>
		<-quiet>: This optional parameter can be added to activate/deactivate detailed logs during the execution of the tool.
gen_msg_doc	Generates the message documentation files	<xsd_source_file>: Source file containing the XML Schema definition of the message</xsd_source_file>
		<xsd_destination_file>:Destination file (full path) in which the flat (without any include) structure definition in XSD format will be saved</xsd_destination_file>
		<activation>: (Optional) File containing the activation configuration for the file being processed</activation>
		<-quiet>: This optional parameter can be added to activate/deactivate detailed logs during the execution of the tool.

gen_sync_map	Generates the message mapping definition files for the Synchronizer	<pre><xsd_source_file>: Source file containing the XML Schema definition of the message <mapping_destination_file>: Destination file in which the mapping generation in the old XML format will be saved -schemaref_dir<dir> (Optional) Directory containing the core message definitions -activation<file>: (Optional) File containing the activation configuration for the file being processed</file></dir></mapping_destination_file></xsd_source_file></pre>
		<-quiet>: (Optional) Parameter that can be added to activate/deactivate detailed logs during the execution of the tool.
gen_oss_map	Generates the message mapping definition files for the Connectors	<pre><xsd_source_file>: Source file containing the XML Schema definition of the message <mapping_destination_file>: Destination file in which the mapping generation in the old XML format will be saved -core_override<file> (Optional) File containing the xml file defining the default mapping -core_override_dir<dir> (Optional) Directory containing the core_override files are located</dir></file></mapping_destination_file></xsd_source_file></pre>
		-custom_override <file<, file="">: (Optional and multiple) File containing the mapping override configuration for this message. Several files can be specified and must be separated with a comma. Each file will be applied in the same order as it is specified. When you use multiple files on Windows platforms, the entire list of files must be enclosed in inverted commas ("file1, file2") <-quiet>: This optional parameter can be added to activate/deactivate detailed logs during the execution of the tool.</file<,>

Connector Generator Tool

This tool generates the following connector runtime files from the Integration Logic Connector Design (<connector name>.ilcd) file:

- Integration Logic Connector Configuration file (<connector name>.ilcc)
- Integration Logic Connector Run time configuration file (<connector name>.ilcr)

ilccGenerator Syntax

ilccGenerator -help | [-check] [-verbose] [-dir <output
directory>] <ILCD file>

PARAMETERS	DESCRIPTION	
help	Displays help for the tool	
check	Checks the validity of the ilcd file before generating	
verbose	Displays all generation messages	
dir <output directory=""></output>	Specifies the destination directory if other than the location of the ilcd file	
<connector_name></connector_name>	The ilcd file to use to generate the connector runtime files	

APPENDIX B

Core Messages

The following table lists available core messages:

MESSAGE NAME	VERB	NOUN
ack.xml	Acknowledge	request
doaddbillingaccount.xml	doadd	billingaccount
doaddcontact.xml	doadd	contact
doaddcontract.xml	doadd	contract
doaddlevel.xml	doadd	level
doaddlogin.xml	doadd	login
doaddmember.xml	doadd	member
doaddorg.xml	doadd	org
doaddservice.xml	doadd	service
doaddtroubleticket.xml	doadd	troubleticket
doassociatededicatedoffer.xml	doassociate	offer
dodeclarepaymentresp.xml	dodeclare	paymentresp
dodissociatededicatedoffer.xml	dodissociate	offer
domigratecontract.xml	domigrate	contract
domodifyaddress.xml	domodify	address
domodifybillingaccount.xml	domodify	billingaccount
domodifycontact.xml	domodify	contact
domodifycontract.xml	domodify	contract
domodifycontractline.xml	domodify	contractline
domodifycontractowner.xml	domodify	contractowner
domodifycontractstatus.xml	domodify	contractstatus
domodifylanguage.xml	domodify	language

domodifylevel.xml domodify member domodifymember.xml domodify org domodifyarymentinfo.xml domodify paymentinfo domodifypaymentinfo.xml domodify rateplan domodifyrateplan.xml domodify service domodifytroubleticket.xml domodify troubleticket domodifytroubleticket.xml domodify troubleticket dossapproval.xml dossapproval approval doremove service doreplaceservice.xml doremove service dosetaddinfo.xml doset addinfo doset personalinfo.xml doset persoinfo notifychange.xml notify change reqaddbillingaccount.xml reqadd bilaccount reqaddbillingaccount.xml reqadd bilaccount reqaddcontract.xml reqadd legalcontact reqaddcontract.xml reqadd level reqaddlegalcontact.xml reqadd login reqaddlegalcontact.xml reqadd org reqaddorg.xml reqadd org reqaddorg.xml reqadd personality org reqaddorg.xml reqadd reqadd requel reqaddorg.xml reqadd service reqaddorg.xml reqadd service reqaddorg.xml reqadd service reqaddorg.xml reqadd service reqaddorg.xml reqadd triblicket reqaddorg.xml reqadd triblicket reqaddorg.xml reqadd triblicket reqaddorg.xml reqadd triblicket reqassociatededicatedoffer.xml reqadd dedicatedoffer reqadsociatededicatedoffer.xml reqadd requested dedicatedoffer reqdeclarepaymentresp.xml reqdeclare reqdissociated dedicatedoffer.xml reqdissociate dedicatedoffer			
domodifyorg.xml domodify org domodifypaymentinfo.xml domodify rateplan domodifyrateplan.xml domodify service domodifyroubleticket.xml domodify troubleticket domodifytroubleticket.xml domodify troubleticket dossapproval.xml dossapproval approval doremoveservice.xml doremove service doreplaceservice.xml doreplace service dosetaddinfo.xml doset addinfo dosetpersonalinfo.xml doset persoinfo notifychange.xml notify change reqaddbillingaccount.xml reqadd bilaccount reqaddbillingaccount.xml reqadd contract reqaddcontract.xml reqadd legalcontac reqaddlevel.xml reqadd level reqaddlevel.xml reqadd login reqaddlogin.xml reqadd member reqaddorg.xml reqadd org reqaddorg.xml reqadd org reqaddorg.xml reqadd service reqaddorg.xml reqadd service reqaddorg.xml reqadd triblticket reqaddservice.xml reqadd service reqaddservice.xml reqadd service reqaddservice.xml reqadd triblticket reqassociatededicatedoffer.xml reqadd dedicatedoffer reqdeclarepaymentresp.xml reqdeclare paymentres	domodifylevel.xml	domodify	level
domodifypaymentinfo.xml domodify rateplan domodifyrateplan.xml domodify service domodifyrorubleticket.xml domodify troubleticket domodifyrorubleticket.xml domodify troubleticket doossapproval.xml dossapproval approval doremoveservice.xml doreplace service doreplaceservice.xml doreplace service dosetaddinfo.xml doset addinfo dosetpersonalinfo.xml doset persoinfo notifychange.xml notify change reqaddbillingaccount.xml reqadd bilaccount reqaddbillingaccount.xml reqadd contract reqaddlegalcontact.xml reqadd legalcontac reqaddlegalcontact.xml reqadd level reqaddlevel.xml reqadd login reqaddlogin.xml reqadd org reqaddlogin.xml reqadd org reqaddorg.xml reqadd org reqaddorg.xml reqadd service reqaddorg.xml reqadd service reqaddorg.xml reqadd trbticket reqaddservice.xml reqadd trbticket reqaddservice.xml reqadd trbticket reqassociatededicatedoffer.xml reqadd paymentres reqadeclarepaymentresp.xml reqdeclare reqdeclarepaymentresp.xml reqdeclare reqdissociatededicatedoffer.xml reqdeclare reqdissociatededicatedoffer.xml reqdeclare reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	domodifymember.xml	domodify	member
domodifyrateplan.xml domodify rateplan domodifyservice.xml domodify service domodifytroubleticket.xml domodify troubleticket doossapproval.xml doossapproval approval doremove service.xml doreplace service doreplaceservice.xml doset addinfo dosetaddinfo.xml doset persoinfo notifychange.xml notify change reqaddbillingaccount.xml reqadd bilaccount reqaddbillingaccount.xml reqadd contract reqaddcontract.xml reqadd level reqaddlegalcontact.xml reqadd level reqaddlevel.xml reqadd login reqaddlogin.xml reqadd org reqaddorg.xml reqadd org reqaddorg.xml reqadd reqadd reqadd reqadd reqaddrey.xml reqadd torg reqaddorg.xml reqadd reqadd reqadd reqadd reqaddrey.xml reqadd reqadd reqadd reqadd reqadd reqadd reqaddrey.xml reqadd service reqaddorg.xml reqadd reqadd trblticket reqaddservice.xml reqadd trblticket reqassociatededicatedoffer.xml reqadsciate dedicatedoffer reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	domodifyorg.xml	domodify	org
domodifyservice.xml domodify troubleticket.xml domodify troubleticket xml domodify troubleticket xml domodify troubleticket xml dossapproval approval approval approval doremoveservice.xml doremove service service doreplaceservice.xml doset addinfo doset addinfo doset addinfo doset personalinfo.xml doset personalinfo.xml doset personalinfo.xml notify change reqaddbillingaccount.xml reqadd bilaccount personalingcontact.xml reqadd bilaccount reqaddcontract.xml reqadd contract reqaddlegalcontact.xml reqadd legalcontact reqaddlevel.xml reqadd level reqaddlevel.xml reqadd member reqadd organia reqadd service xml reqadd service xml reqadd tribiticket reqadderepaymentresp.xml reqadd dedicatedoffer paymentres addicatedoffer.xml reqdeclare paymentres	domodifypaymentinfo.xml	domodify	paymentinfo
domodifytroubleticket.xml domodify troubleticket doossapproval.xml dossapproval approval doremoveservice.xml doremove service dosetaddinfo.xml doset addinfo dosetpaceservice.xml doset persoinfo notifychange.xml notify change reqaddbillingaccount.xml reqadd bilaccount reqaddbillingcontact.xml reqadd bilaccount reqaddlegalcontact.xml reqadd legalcontac reqaddlegalcontact.xml reqadd login reqaddlogin.xml reqadd reqadd login reqaddlogin.xml reqadd reqadd reqadd reqaddlogin.xml reqadd reqadd reqaddlogin.xml reqadd reqadd reqaddlogin.xml reqadd reqaddlogin.xml reqadd reqaddlogin.xml reqadd reqadd reqaddlogin.xml reqadd reqadd reqaddlogin.xml reqaddlogin.xml reqadd reqaddlogin.xml reqaddlogin	domodifyrateplan.xml	domodify	rateplan
doossapproval.xml doossapproval approval approval doremoveservice.xml doremove service service doreplaceservice.xml doreplace service addinfo.xml doset addinfo doset persoinfo notifychange.xml notify change reqaddbillingaccount.xml reqadd bilaccount preqaddbillingaccount.xml reqadd bilaccount reqaddcontract.xml reqadd contract reqaddlegalcontact.xml reqadd legalcontact reqaddlegalcontact.xml reqadd legalcontac reqaddlegalcontact.xml reqadd reqaddlogin.xml reqadd reqadd login reqaddlogin.xml reqadd org reqaddlogontact.xml reqadd reqadd org reqaddlogontact.xml reqadd reqadd reqaddlogontact.xml reqadd reqadd reqaddlogontact.xml reqadd reqadd reqaddlogontact.xml reqadd reqadd reqadd reqaddlogontact.xml reqadd reqadd reqadd reqaddlogontact.xml reqadd reqadd reqadd reqaddlogontact.xml reqadd reqadd service reqaddroubleticket.xml reqadd trblticket reqassociatededicatedoffer.xml reqaddlegelcatedoffer paymentres reqdeclare paymentres reqdeclare paymentres reqdeclared reqdissociate dedicatedoffer reqdeclared reqdissociate dedicatedoffer reqdeclared reqdissociate dedicatedoffer reqdeclared reqdissociate dedicatedoffer reqdissociate reqdissociate reqdissociated required reqdissociated required reqdisso	domodifyservice.xml	domodify	service
doremoveservice.xml doremove service doreplaceservice.xml doreplace service dosetaddinfo.xml doset addinfo dosetpersonalinfo.xml doset persoinfo notifychange.xml notify change reqaddbillingaccount.xml reqadd bilaccount reqaddbillingcontact.xml reqadd bilcontact reqaddlegalcontact.xml reqadd legalcontac reqaddlegalcontact.xml reqadd level reqaddlevel.xml reqadd login reqaddlogin.xml reqadd org reqaddmember.xml reqadd member reqaddorg.xml reqadd org reqaddorg.xml reqadd service reqaddservice.xml reqadd service reqaddservice.xml reqadd trblticket reqaddscontact.xml reqadd personact reqaddorg.xml reqadd personact reqaddorg.xml reqadd trblticket reqaddservice.xml reqadd trblticket reqaddscontact.xml reqadd trblticket reqadscociatededicatedoffer.xml reqadd paymentres reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	domodifytroubleticket.xml	domodify	troubleticket
doreplaceservice.xml doset addinfo doset addinfo dosetpersonalinfo.xml doset persoinfo notifychange.xml notify change reqaddbillingaccount.xml reqadd bilaccount reqaddbillingcontact.xml reqadd bilcontact reqaddlegalcontact.xml reqadd legalcontac reqaddlegalcontact.xml reqadd level reqaddlogin.xml reqadd login reqaddmember.xml reqadd org reqaddorg.xml reqadd reqadd org reqaddrog.xml reqadd service reqaddsevel.xml reqadd togin reqaddrog.xml reqadd togin reqadd togin reqaddrog.xml reqadd togin reqadd togin reqaddorg.xml reqadd porgontact reqaddorg.xml reqadd porgontact reqaddservice.xml reqadd togin reqaddservice.xml reqadd porgontact reqaddroubleticket.xml reqadd porgontact reqadsociatededicatedoffer.xml reqadd porgontact reqassociatededicatedoffer.xml reqassociate dedicatedoffer	doossapproval.xml	doossapproval	approval
dosetaddinfo.xml doset addinfo doset persoinfo notifychange.xml notify change reqaddbillingaccount.xml reqadd bilaccount reqaddbillingcontact.xml reqadd contract reqaddlegalcontact.xml reqadd legalcontac reqaddleyel.xml reqadd level reqaddlogin.xml reqadd login reqaddmember.xml reqadd org reqaddorg.xml reqadd org reqadd org reqaddorg.xml reqadd treblicket.xml reqadd tredadd org reqaddorg.xml reqadd triblicket reqaddservice.xml reqadd trblicket reqaddtroubleticket.xml reqadd trblicket reqaddtroubleticket.xml reqadd tredaddere reqaddcarepaymentresp.xml reqadd dedicatedoffer reqdeclarepaymentresp.xml reqdeclare reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	doremoveservice.xml	doremove	service
dosetpersonalinfo.xml doset persoinfo notifychange.xml notify change reqaddbillingaccount.xml reqadd bilaccount reqaddbillingcontact.xml reqadd bilcontact reqaddcontract.xml reqadd contract reqaddlegalcontact.xml reqadd legalcontac reqaddleyel.xml reqadd login reqaddlogin.xml reqadd login reqaddmember.xml reqadd member reqaddorg.xml reqadd org reqadd org reqaddorg.xml reqadd torg reqaddservice.xml reqadd torg reqadd torg reqaddservice.xml reqadd reqadd reqadd reqadd reqaddservice.xml reqadd service reqaddstroubleticket.xml reqadd triblicket reqassociatededicatedoffer.xml reqadd dedicatedoffer reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	doreplaceservice.xml	doreplace	service
reqaddbillingaccount.xml reqadd bilaccount reqaddbillingcontact.xml reqadd bilcontact reqaddcontract.xml reqadd contract reqaddlegalcontact.xml reqadd legalcontac reqaddlevel.xml reqadd level reqaddlogin.xml reqadd login reqaddmember.xml reqadd org reqaddorg.xml reqadd org reqaddorg.xml reqadd tredadd org reqaddorg.xml reqadd tredadd org reqaddorg.xml reqadd tredadd org reqaddorg.xml reqadd tredadd reqadd reqadd reqadd org reqaddorg.xml reqadd tredadd service reqaddservice.xml reqadd trblticket reqaddtroubleticket.xml reqadd trblticket reqassociatededicatedoffer.xml reqadd requires reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	dosetaddinfo.xml	doset	addinfo
reqaddbillingaccount.xml reqadd bilaccount reqaddbillingcontact.xml reqadd bilcontact reqaddcontract.xml reqadd contract reqaddlegalcontact.xml reqadd legalcontac reqaddlevel.xml reqadd level reqaddlogin.xml reqadd login reqaddmember.xml reqadd org reqaddorg.xml reqadd org reqaddorg.xml reqadd torg reqaddorg.xml reqadd torg reqaddorg.xml reqadd torg reqaddorg.xml reqadd torg reqaddservice.xml reqadd service reqaddservice.xml reqadd trblticket reqassociatededicatedoffer.xml reqadd trblticket reqassociatededicatedoffer.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	dosetpersonalinfo.xml	doset	persoinfo
reqaddbillingcontact.xml reqadd bilcontact reqaddcontract.xml reqadd contract reqaddlegalcontact.xml reqadd legalcontac reqaddleyel.xml reqadd level reqaddlogin.xml reqadd login reqaddmember.xml reqadd member reqaddorg.xml reqadd org reqaddorg.xml reqadd org reqaddservice.xml reqadd service reqaddservice.xml reqadd trblticket reqaddtroubleticket.xml reqadd trblticket reqassociatededicatedoffer.xml reqadd dedicatedoffer reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	notifychange.xml	notify	change
reqaddcontract.xml reqadd contract reqaddlegalcontact.xml reqadd legalcontac reqaddlevel.xml reqadd level reqaddlogin.xml reqadd login reqaddmember.xml reqadd member reqaddorg.xml reqadd org reqaddorg.xml reqadd org reqaddservice.xml reqadd service reqaddservice.xml reqadd trblticket reqaddtroubleticket.xml reqadd trblticket reqassociatededicatedoffer.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	reqaddbillingaccount.xml	reqadd	bilaccount
reqaddlegalcontact.xml reqadd legalcontac reqaddlevel.xml reqadd level reqaddlogin.xml reqadd login reqaddmember.xml reqadd member reqaddorg.xml reqadd org reqaddorg.xml reqadd org reqaddservice.xml reqadd service reqaddtroubleticket.xml reqadd trblticket reqassociatededicatedoffer.xml reqdeclare reqdeclarepaymentresp.xml reqdeclare reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	reqaddbillingcontact.xml	reqadd	bilcontact
reqaddlevel.xml reqadd login reqadddrounter.xml reqadd member reqaddorg.xml reqadd org reqaddorg.xml reqadd org reqaddorg.xml reqadd org reqaddservice.xml reqadd service reqaddservice.xml reqadd trblticket reqaddtroubleticket.xml reqadd trblticket reqassociatededicatedoffer.xml reqadd dedicatedoffer reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	reqaddcontract.xml	reqadd	contract
reqaddlogin.xml reqadd login reqaddmember.xml reqadd member reqaddorg.xml reqadd org reqaddorg.xml reqadd org reqaddservice.xml reqadd service reqaddtroubleticket.xml reqadd trblticket reqassociatededicatedoffer.xml reqassociate dedicatedoffer reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	reqaddlegalcontact.xml	reqadd	legalcontac
reqaddmember.xml reqadd member reqaddorg.xml reqadd org reqaddorgcontact.xml reqadd orgcontact reqaddservice.xml reqadd service reqaddtroubleticket.xml reqadd trblticket reqassociatededicatedoffer.xml reqassociate dedicatedoffer reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	reqaddlevel.xml	reqadd	level
reqaddorg.xml reqadd org reqaddorgcontact.xml reqadd orgcontact reqaddservice.xml reqadd service reqaddtroubleticket.xml reqadd trblticket reqassociatededicatedoffer.xml reqassociate dedicatedoffer reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	reqaddlogin.xml	reqadd	login
reqaddorgcontact.xml reqadd orgcontact reqaddservice.xml reqadd service reqaddtroubleticket.xml reqadd trblticket reqassociatededicatedoffer.xml reqassociate dedicatedoffer reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	reqaddmember.xml	reqadd	member
reqaddservice.xml reqadd service reqaddtroubleticket.xml reqadd trblticket reqassociatededicatedoffer.xml reqassociate dedicatedoffer reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	reqaddorg.xml	reqadd	org
reqaddtroubleticket.xml reqadd trblticket reqassociatededicatedoffer.xml reqassociate dedicatedoffer reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	reqaddorgcontact.xml	reqadd	orgcontact
reqassociatededicatedoffer.xml reqassociate dedicatedoffer reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	reqaddservice.xml	reqadd	service
reqdeclarepaymentresp.xml reqdeclare paymentres reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	reqaddtroubleticket.xml	reqadd	trblticket
reqdissociatededicatedoffer.xml reqdissociate dedicatedoffer	reqassociatededicatedoffer.xml	reqassociate	dedicatedoffer
	reqdeclarepaymentresp.xml	reqdeclare	paymentres
roadestan yed	reqdissociatededicatedoffer.xml	reqdissociate	dedicatedoffer
requestaeciaration.xrni requestare lostπieπ	reqlostdeclaration.xml	reqdeclare	lostthieft

and the second second second		
reqmigratecontract.xml	reqmigrate	contract
reqmodifybillingaccount.xml	reqmodify	billingaccount
reqmodifybillingaddress.xml	reqmodify	biladdress
reqmodifybillingcontact.xml	reqmodify	bilcontact
reqmodifycontract.xml	reqmodify	contract
reqmodifycontractline.xml	reqmodify	contractline
reqmodifycontractowner.xml	reqmodify	contractowner
reqmodifycontractstatus.xml	reqmodify	contractstatus
reqmodifylanguage.xml	reqmodify	language
reqmodifylegalcontact.xml	reqmodify	legalcontact
reqmodifylevel.xml	reqmodify	level
reqmodifymember.xml	reqmodify	member
reqmodifyorg.xml	reqmodify	org
reqmodifyorgaddress.xml	reqmodify	orgaddress
reqmodifyorgcontact.xml	reqmodify	orgcontact
reqmodifypaymentinfo.xml	reqmodify	paymentinfo
reqmodifyrateplan.xml	reqmodify	rateplan
reqmodifyservice.xml	reqmodify	service
reqmodifytroubleticket.xml	reqmodify	trblticket
reqorder.xml	reqorder	composite
reqorderdoc.xml	reqorder	doc
reqrechargeprepaid.xml	reqrecharge	prepaid
regremoveservice.xml	reqremove	service
reqreplaceservice.xml	reqreplace	service
reqsetpersonalinfo.xml	reqset	persoinfo
system.xml	system	system

APPENDIX C

Core Business Objects

In This Section

About Core Business Objects	208
About Parameters and Messages	209

About Core Business Objects

This list presents the core Business objects. It is a list of the Business Objects which support additional parameters:

- BillingAccount
- Organization
- Organization category type
- Language
- Role
- Level
- Member
- Service
- Contract
- Rate plan
- Line
- Line Type
- Country
- Title
- Sex
- Contact
- Trouble ticket
- Trouble Ticket category
- Document
- Invoice
- Prepaid package
- Commercial Offer

About Parameters and Messages

The SmartLink (ISF) is configured to manage the additional parameters of every object that can have additional information.

Outbound Messages

If the object is on the list of supported objects, the SmartLink (ISF) is ready to handle additional parameters. If the object can support additional parameters, but has this feature inactivated, refer to Working With the Message Schema Reference.

Inbound Messages

The inbound messages come with a special section that deals with the additional information of all supported objects. In a message, this dedicated section manages updating all additional information.

The additional information in this section does not necessarily need to be related to the other objects being modified. This feature allows you to update additional information of unrelated objects when your application requires an update in a single transaction.

The doSetAddInfo message contains modifications to the additional parameters of each object that supports additional information. This message has a field for each object that supports additional information and is used to update additional information only.

APPENDIX D

Element Reference

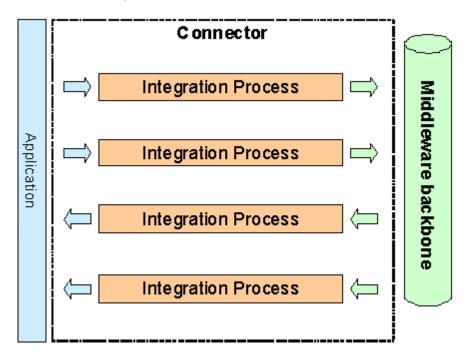
In This Section

Processors	214
Message Queues	215
Request Queue	226
Utilities	229
Custom Processors	232
Routers	233
Extensions	244
Parameters	245

Integration Processes

An Integration Process is the dataflow for requests or messages between a system and the transport layer, or between the transport layer and one or more systems. Each connector includes one or more Integration Processes.

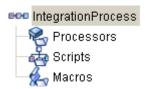
These processes consist of a number of processors connected together in a sequence to handle messages.



An integration process has the following:

- One or more Processors
- One or more Scripts
- One or more Macros

They are represented by the following icon:



The components of the integration process carry out the following basic tasks:

1 Extract Data

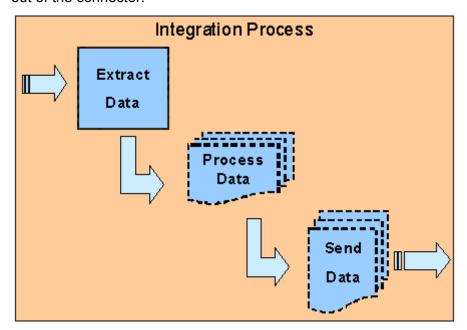
This task is extracting the data the integration process works on. This is a mandatory task and is carried out by a unique processor.

2 Process Data

This task is processing the extracted data. This is also mandatory and has several different components which handle data, forwards data to another processor, generates messages, handles errors, routes messages and so on.

3 Sending Data

This optional tasks is at the end of the processing and involves sending the data out of the connector.



Processors

A Processor is an element that handles or transforms a message.

The basic principle of a processor is:

- Receive a message
- Carry out tasks

Tasks include:

- Processing the content of the message
- Activating scripts or macros
- Routing the message

You link processors together to create the message flow of your connector. Depending on the processor, possible links include:

- Send connections
- Error connections
- Exception connections

This diagram illustrates how processors create an Integration Process workflow:

Integration Process Processor 4 Processor 8 Processor 1 Processor 2 Send message Filter Messages whose type is to middleware not queue 1 "AddService" Extract a Duplicate message the message fromthe midd leware Processor 7 Processor 6 Filter Messages Send message to middleware whose type is queue 2 not "AddContract"

Message Queues

The processors in this category are for message queues. A message queue is a temporary storage location where messages are held.

The types of queues are:

- Inbound
 - These queues are for messages that the connector receives for processing.
- Outbound
 - These queues are for messages that the connector has processed and are ready to be sent.
- Persistence

These queues are for messages that have been saved in a database/

Queues also have an associated transport protocol. The available transport protocols include:

- Email
- File
- Socket
- JMS

Inbound Batch Queue

This message queue is a queue that receives inbound messages.

The messages handled in this queue are files containing file lists. Each message has a single list of files. Each file on the list is either a file or archive. For each listed file or archive, this queue does the following:

- For files, it creates a message containing the reference to the file
- For archives, it opens the archive. For each file in the archive, it creates a message containing the reference to the file

The inbound batch queue can handle gzip and zip file compression formats only.

This element has the following properties:

PROPERTY	DESCRIPTION
Directory Name	Full path of the directory to scan for the files
Prefix	Prefix of file list names
Message Type	Message type to create
Msg Count Variable	Variable to use with the Message Counter Processor
Napping Time	Time between polling (in milliseconds)
Number of threads	Number of threads to allocate
Java priority	JVM priority of the threads

Inbound Email Queue

This message queue is a queue that receives inbound messages.

The inbound messages in this queue are transmitted using emails. The messages come from a mail server using the POP3 protocol.

This element has the following properties:

PROPERTY	DESCRIPTION
Host name	Hostname of the POP server and the domain (example: myhost. mydomain.com)
User Name	POP account name
User Password	POP account password
Napping time	Time between polling (in milliseconds)
Number of threads	Number of threads to allocate
Java priority	JVM priority of the threads

Inbound File System Queue

This message queue is a queue for inbound messages.

The inbound messages in this queue are transmitted using the file system.

PROPERTY	DESCRIPTION
Directory	Path of the file system queue
Napping time	Time between polling (in milliseconds)
Number of threads	Number of threads to allocate
Java priority	JVM priority of the threads

Inbound JMS Queue

This message queue is a queue that receives inbound messages.

The inbound messages in this queue are transmitted using JMS point-to-point queues.

PROPERTY	DESCRIPTION
JMS Queue Name	Queue name obtained by JNDI
Connection Factory	JMS connection factory obtained through JNDI
Transactional Flag	Boolean. If set to true, creates a transaction for each message
Napping time	Time between polling (in milliseconds)
Number of threads	Number of threads to allocate
Java priority	JVM priority of the threads

Inbound JMS Topic Queue

This message queue is a queue that receives inbound messages.

The inbound messages in this queue are transmitted using JMS publish/subscribe queues.

This element has the following properties:

PROPERTY	DESCRIPTION
Connection Factory	JMS connection factory obtained through JNDI
Transactional Flag	Boolean. If set to true, creates a transaction for each message
Napping time	Time between polling (in milliseconds)
Number of threads	Number of threads to allocate
Java priority	JVM priority of the threads

This element one or more JMS Topic Names.

Each JMS Topic Name has the following properties:

PROPERTY	DESCRIPTION
----------	-------------

Topic name

Inbound Memory Queue

This message queue is a queue for inbound messages.

The inbound messages in this queue are transmitted using allocated system memory.

This element has the following properties:

PROPERTY	DESCRIPTION
Queue Name	Name of the memory queue
Number of threads	Number of threads to allocate
Java priority	JVM priority of the threads

Inbound Socket Queue

This message queue is a queue that receives inbound messages.

The inbound messages in this queue are transmitted using TCP/IP sockets.

PROPERTY	DESCRIPTION
Host Name	Host name
Port Number	The TCP/IP socket to connect to
Number of threads	Number of threads to allocate
Java priority	JVM priority of the threads

Outbound Email Queue

This message queue is a queue for outbound messages.

The outbound messages in this queue are transmitted using emails. The messages are sent to a mail server using the SMTP protocol.

This element has the following properties:

PROPERTY	DESCRIPTION
Host name	Hostname of the POP server and the domain (example: myhost. mydomain.com)
User Name	POP account name
User Password	POP account password
Recipient	Email address of the recipient (example: user@mydomain.com)
Xmailer	Xmailer

Outbound File System Queue

This message queue is a queue for outbound messages.

The outbound messages in this queue are transmitted using the file system.

PROPERTY	DESCRIPTION
Directory name	Path of the file system queue

Outbound JMS Queue

This message queue is a queue for outbound messages.

The outbound messages in this queue are transmitted using JMS point-to-point queues.

This element has the following properties:

PROPERTY	DESCRIPTION
JMS Queue Name	Queue name obtained by JNDI
Connection Factory	JMS connection factory obtained through JNDI
Transactional Flag	Boolean. If set to true, creates a transaction for each message

Outbound JMS Topic Queue

This message queue is a queue for outbound messages.

The outbound messages in this queue are transmitted using JMS publish/subscribe queues.

This element has the following properties:

PROPERTY	DESCRIPTION
JMS Connection factory	JMS connection factory obtained through JNDI
Transaction flag	Boolean. If set to true, creates a transaction for each message

This element one or more JMS Topic Names.

Each JMS Topic Name has the following properties:

PROPERTY	DESCRIPTION

Topic name

Outbound Memory Queue

This message queue is a queue for outbound messages.

The inbound messages in this queue are transmitted using allocated system memory.

This element has the following properties:

PROPERTY	DESCRIPTION
Queue Name	Name of the memory queue
Time Out	Maximum delay for a put operation (in milliseconds)
Queue Size	The maximum number of messages that the queue can hold.

Outbound Socket Queue

This message queue is a queue for outbound messages.

The outbound messages in this queue are transmitted using TCP/IP sockets.

This element has the following properties:

PROPERTY	DESCRIPTION
Host Name	Host name
Port Number	The TCP/IP socket to connect to

Message Cache Polling Processor

This message queue is a queue for messages which have been saved in the SmartLink (ISF) Message Cache.

This processor does the following:

- 4 Extracts data from the configured message cache at a specified interval
- 5 Creates a new message whose body will contain extracted data
- 6 Send this message to the next connected processor in the integration process

This element has the following properties:

PROPERTY	DESCRIPTION
DB User Name	Name to connect to the database
DB User Password	Corresponding password
DB Connection String	Connection string
DB Driver	Database driver
Cache Name	Name of the message cache
Number of threads	Number of threads to allocate
Napping time	Time between polling (in milliseconds)
Max No. Of Messages	Maximum number of messages to extract each time
Custom Extraction SQL	Path to the file containing custom SQL script
User exits class	Class containing user exit code to execute
Unread Messages Only	Boolean. If set to true, extract only unread messages
Save Error Description	Boolean. If set to true, saves the description of any error.

Database Polling Processor

This message queue is a queue for messages which have been saved in the SmartLink (ISF) Message Cache.

This processor does the following:

- 7 Extracts at a periodic interval data using specified SQL statement
- 8 Creates a new message whose body will contain:
 - Either extracted data
 - Or the result of a user provided message factory. This factory will take as input parameter the extracted data and will output a standard message with type, body and envelop.
- **9** Send this message to the next connected processor in the integration process

PROPERTY	DESCRIPTION
DB User Name	Name to connect to the database
DB User Password	Corresponding password
DB Connection String	Connection string
Number of threads	Number of threads to allocate
Napping time	Time between polling (in milliseconds)
Flag Read Line	Boolean. If set to true, flags the row as being read
Delete Row On Success	Boolean. If set to true, deletes the row when the extraction occurs without errors
Save Error Description	Boolean. If set to true, saves the description of any error.
User exits class	Class containing user exit code to execute
Identifier Column Name	Name of the column containing the ID name
Identifier Column Type	Type of the column containing the ID
Selection Query	File name of the code used to select messages
Lock Query	File name of the code used to lock messages
Message Factory Class	Class containing the message factory
Confirm Update	
Confirm Error	
Confirm Error With Msg	

Request Queue

The processors in this category are for managing messages extracted from the request queue.

The request queue is the table in the CID which holds the requests of the TSM. The Request Queue Agent polls this table and extracts messages from this table for processing.

Request Queue Agent

This Request Queue Agent element is the agent that manages the extraction of data from the CID.

This element has the following properties:

PROPERTY	DESCRIPTION
Туре	Type of element
	DO NOT MODIFY
Number of Threads	Number of threads to allocate
Napping Time	Time between polling (in milliseconds)
Looping Time	Time between repeats (in milliseconds)
Synchronous Only	Extract Synchronous Requests Only (boolean)
Execute Once	Oneshot Request Queue Agent (boolean)
No. of Requests	Number of requests extracted
Internal Queue Size	Size of the internal queue
Reconnection Delay	Time to wait before trying to reconnect to the CID
Reconnection Retries	Number of times to try to reconnect to the CID
BLM User Name	User name to use to connect ot the BLM
BLM Password	Associated password

This element contains:

One or more Request Types

The *Request Types* correspond to the request types the Request Queue Agent processes.

PROPERTY	DESCRIPTION
Enabled	Boolean
Execution Type	
BLM Request Type	
Message Type	

Each request type can have one *Status Codes* element.

The Status Codes element has following properties:

PROPERTY	DESCRIPTION
Success Status	Boolean
Failure Status	
Transport Retry Status	
Transport Failure Status	
Waiting for ACK Status	
Acknowledged Status	

One Sequence

The Sequence is ...

The Sequence element has the following properties:

PROPERTY	DESCRIPTION
Enabled	Boolean

Utilities

The processors in this category are utilities for message processing and flow.

Encoding Converter

The Encoding Converter element specifies the encoding of messages.

This element has the following properties:

PROPERTY	DESCRIPTION
Middleware Encoding	The name of the character set to use for the messages
XML Encoding	The name of the character set to use for the XML files

DOM Builder

The DOM Builder converts a XML document into DOM document.

This element has no properties.

DOM Serializer

The DOM Serializer converts a DOM document into an XML document.

This element has no properties.

XML Validator

The XML Validator element validates the XML document using a specified schema.

This element has the following properties:

PROPERTY	DESCRIPTION
Validation Flag	Boolean. If set to true, validates the XML document using the specified format and file
Schema Format	Schema formats available via the CLASSPATH to use for validation.
	To add a format, enter the Validator to the CLASSPATH.
Schema File	Full path of the Schema file to use.

Message Counter

The Message Counter processor count messages.

This element has variables which must be defined in your Connector macro files.

PROPERTY	DESCRIPTION
Msg Count Variable	The name of the variable to use for counting messages.
Success Count Variable	The name of the variable to hold the value of successful transmissions
Exception Count Variable	The name of the variable to hold the value of unsuccessful transmissions

Pause

The Pause element specifies the delay between two actions.

PROPERTY	DESCRIPTION
Delay	The time to wait between two actions (in milliseconds)

Custom Processors

The processors in this category are the processors you have created for your connector. They are listed here to ease reuse of your custom processors.

Routers

The processors in this category are for broadcasting messages and managing message flows.

In general, Messages are broadcast to the queues and the queue has an associated filter to determine whether or not the queue accepts the message. There are processors in this category to manage the broadcasting and filtering of messages for a single queue or multiple queue configurations.

Script Engine

A script engine is in charge of running scripts which manage the routing of data.

This element has no properties.

This element contains:

One or more Script name elements

PROPERTY	DESCRIPTION
Script name	Name of the script

Message Type Filter

The message type filter is a list of message types that are used to filter broadcast messages.

This element has no properties.

This element contains:

One or more Message Type elements

The *Message Type* correspond to the message types this filter passes on to the queue. All other messages are ignored.

PROPERTY	DESCRIPTION
Message Type	Name of the message type

Message Broadcaster

The message broadcaster sends messages to the queue.

The queue has a message filter which lists the messages that are placed in the queue. All other messages are ignored.

This element has the following properties:

PROPERTY	DESCRIPTION
Duplicate Flows	Boolean. When set to true, the Message Broadcaster duplicates the message before sending it.

MultiQueue Message Filter

A message type filter for multi queue configurations.

This element has no properties.

This element contains:

One or more Message Type elements

The *Message Type* correspond to the message types this filter passes on to the queue. All other messages are ignored.

PROPERTY	DESCRIPTION
Message Type	Name of the message type

MultiQueue Message Broadcaster

This message broadcaster is for multiqueue configurations.

The broadcaster sends messages to all of the queues. The queues have message filters which list the messages that are placed in the queue.

This element has the following properties:

PROPERTY	DESCRIPTION
Database Driver	The name of the database driver to use
Database URL	The location of the CID database
Database User Name	The user name to connect to the database
Database Password	The associated password

MultiQueue Message Tracker

This message tracker is for multiqueue configurations.

The message tracker manages the dispatching of messages to different queues.

This element has the following properties:

PROPERTY	DESCRIPTION
Database Driver	The name of the database driver to use
Database URL	The location of the CID database
Database User Name	The user name to connect to the database
Database Password	The associated password

The processors in this category are for handling errors. When processing messages, errors may occur and your connector needs to be able to manage them.

For instance, if an error occurs when processing a message, you can configure your connector to retry the message. You can also specify the conversion of an error into another in order to customize your error handling or trigger other events.

Error Handler

An Error Handler is in charge of handling exceptions.

This element has no properties.

Exception Converter

The exception converter converts exceptions from one exception to another.

This element has no properties.

Retry Handler

A Retry Handler is in charge of handling retries when an error occurs.

This element has the following properties:

PROPERTY	DESCRIPTION
Number of Repeats	The number of attempts to send the message
Delay between Repeats	The time between attempts (in milliseconds)

CID Update Error Handler

A CID Update Error Handler handles errors which occur when processing requests that update the CID.

PROPERTY	DESCRIPTION
Error string	The error message

The processors in this category are involved in changing the formats of messages. A Transmapper is a processor element that handles the transformation of a message from one format to another format.

For example, the connector receives a message in the form of a BLM object. This message needs to be transformed into an XML format the connector can work with. The BLM/XML Transmapper handles converting a BLM object into XML.

Transmappers may also have associated files that specify the structure of the output message. They may also have files that manage the mapping of the input and output messages. The BLM/XML Transmapper has both of these files to define the structure of the XML and the mapping of BLM objects to XML elements.

BLM/XML Transmapper

The BLM/XML Transmapper Processor transforms a Request object into an XML string.

This Processor will normally follow immediately after the Request Queue Agent Processor, transforming the minimal information held in the Request object into a complete XML message providing all data required by the target system(s). The specification for this information both in terms of the XML schema and the BLM APIs required to extract the data from the CID are defined in the Message Schema Reference.

This element has the following properties:

PROPERTY	DESCRIPTION
Туре	Type of element DO NOT MODIFY

This transmapper contains:

One or more Request Types

The *Request Types* correspond to the message types this Transmapper processes.

PROPERTY	DESCRIPTION
Structure definition file path	Full path of the XML file containing the definition of the message structure
Mapping definition file path	Full path of the XML file containing the definition of the mapping of BLM content to XML.

Java/XML Transmapper

This transmapper transforms messages from a Java object format into an XML format.

The Java/XML Transmapper Processor is the more generic form of the BLM/XML Transmapper. The Java/XML Transmapper Processor transforms any Java object into an XML string. This processor will often be used within a BSS/OSS Connector to transform BSS/OSS objects into an XML message for delivery.

This element has the following properties:

PROPERTY	DESCRIPTION
Buffered Output	
Output Mode	
Output Format	

This transmapper contains:

One or more Request Types

The *Request Types* correspond to the message types this Transmapper processes.

PROPERTY	DESCRIPTION
Structure definition file path	Full path of the XML file containing the definition of the message structure
Mapping definition file path	Full path of the XML file containing the definition of the mapping of BLM content to XML.

XML Parser

This transmapper transforms messages from an XML format into a recursive hash table format.

This processor is typically used immediately prior to the CID Objects Builder processor, transforming the inbound XML message into a Java hash table as the correct input format for the CID Objects Builder.

This element has the following properties:

PROPERTY	DESCRIPTION
Туре	Type of element DO NOT MODIFY

This transmapper contains:

One or more Request Types

The Request Types correspond to the message types this Transmapper processes.

PROPERTY	
Structure definition file path	Full path of the XML file containing the definition of the message structure

XSL Transmapper

This transmapper applies the specified XSL to the message.

This element has the following properties:

PROPERTY	DESCRIPTION
Туре	Type of element DO NOT MODIFY
Buffered Output	Boolean. If set to true, the output is buffered. The transmapper applies all of the XSLs to the message then sends the results. If false, after each transformation, the transmapper sends the result.
Output Mode	The output mode for the transmapped message:
	- single create a single message
	- multiple create multiple messages
Output Format	The format of the output:
	- DOM Format
	- String Format

This transmapper contains:

• One or more *Transformations*

The *Transformations* correspond to a type of transformation to carry out.

The *Transformations* elements have the following properties:

PROPERTY	DESCRIPTION
Stylesheet	Full path of the XSL file containing the style definitions
Output Message Type	The message type (For instance, REQADDSERVICE)

CID Objects Builder

This transmapper transforms messages from recursive hash table format into Data Access Layer (DAL) objects.

The DAL object are for updating the CID. This processor itself does not update the CID. You use the Script Engine to update the CID.

This element has the following properties:

PROPERTY	DESCRIPTION
Туре	Type of element DO NOT MODIFY

This transmapper contains:

One or more Request Types

The Request Types correspond to the message types this Transmapper processes.

PROPERTY	DESCRIPTION
Structure definition file path	Path to the file containing the message structure for this request type

Macros

A Macro element defines the custom APIs that can be called in scripts.

When using Macros to create your own API, you can:

- Wrap external Java code to custom script APIs
- Instantiate external Java classes

For instance, your connector may have to establish a connection with a back end system when the connector starts up and then reuse this connection for all Integration Processes. In this case, you can use a Macro to instantiate the connection then reuse it whenever necessary in any of your Integration Processes.

A macro can be one of the following:

Connector Macro

These macros can be used by all of the processors. They are located under <connector_home>/Macros.

Integration Process

These macros can only be used by the elements of the Integration Process. They are located under <connector home>/<integration process>/Macros.

Scripts

This element points to an XML file. This XML file is a Script.

A Script is a sequence of commands or operations, which are executed by a Script Engine Processor. Each command actually calls an API, either defined within the SmartLink (ISF) or registered as an external Macro.

A number of core Script features include a Switch command which allows the script to branch conditionally on the results of any variable, providing "If-Then-Else" functionality within the script. Scripts can also execute sub-script components

They are represented by the following icon:



Because of the power and flexibility of Scripts, they are used for a number of functions:

- Updating data in the CID
- Routing messages within an Integration Process
- Calling external APIs for data processing

Extensions

An Extension is an element that customizes the queues.

The extension represents a new queue along with a set of custom parameters. When you add this custom queue to your processor, you enter the values of the custom parameters.

This element has no default properties. You add your parameters to this element.

This element has:

One or more Parameters
 The *Properties* correspond to your custom parameters for the queue.

PROPERTY	DESCRIPTION
parameter name	name of the parameter

Parameters

A Parameter element is a variable you can use when setting the properties of the elements in your connector.

For instance, when developing your connector, you may use a local path for developing and testing. However, when you generate your configuration for deployment, you need to have different paths. If you do not use Parameter elements, you have to manually change the path in all of the elements of your connector.

Parameters can be used to replace values such as:

Paths:

```
path/subpath1/
./path
../../path
```

Strings:

```
dest@company.com
BlueConnectorBeta
```

- Booleans
- Numeric values

Parameter names have the following restrictions:

- No spaces (underscores are allowed)
- Only letters (no numbers or symbols can be used)
- Upper case only

Example of a valid parameter names:

- HOME PATH
- EMAIL
- CONNECTOR NAME

This element has no properties.

This element contains:

• One or more *Parameter* elements

The parameter element specifies the properties of the parameter.

PROPERTY	DESCRIPTION
parameterdescription	Description of the parameter
valuedescription	Description of the value
typedescription	Choose one of the following:
	• string
	• boolean
	• numeric
multiplicitydescription	Choose one of the following:
	• simple
	• multiple
Value	Value of the parameter
	When multiple, parameters can have more than one value property

Index

A	D
Additional Parameters list of objects • 207	DOM Builder about • 229
Administration tools	configuring • 229
agentadm • 183, 192	properties • 229 DOM Serializer
agentstart • 183, 191 ossadmin • 184	
ossstart • 184	about • 229 configuring • 229
agentadm Administration Tool	properties • 229
syntax • 192	properties - 22)
using • 183	E
agentstart Administration Tool	Encoding Converter
syntax • 191	about • 229
using • 183	properties • 229
_	Errors
В	about • 235
BLM/XML Transmapper	CID Update Error Handler • 236
properties • 237	Error Handler • 236
C	Exception Converter • 236 Retry Handler • 236
CID Objects Builder	Exception Converter
properties • 241	about • 236
CID Update Error Handler	properties • 236
about • 236	Extensions
properties • 236	sample inbound queue • 126
Connecting	sample outbound queue • 125
connecting • 139	
Connector Configuration File (ilcc)	l
generating • 176	Inbound Batch Queue
Connector Design File (ilcd)	about • 216
generating • 176	properties • 216
Connector Parameters File (ilcr)	Inbound Email Queue
generating • 176	about • 217
Connectors	properties • 217
administrating • 185	Inbound File System Queue
and Integration Processes • 212	about • 217
and Macros • 242	properties • 217
and Scripts • 243	Inbound JMS Queue
generating • 176	about • 218
running • 183	properties • 218
running OSS Connector • 184	Inbound JMS Topic Oueue

Inbound Email Queue • 217
Inbound File System Queue • 217
Inbound JMS Queue • 218
Inbound JMS Topic Queue • 219
Inbound Socket Queue • 220
Outbound Email Queue • 221
Outbound File System Queue • 221
Outbound JMS Queue • 222
Outbound Socket Queue • 223
Message Schema Reference
about • 148
Message Type Filter
properties • 233
messagecacheAdminTool Administration
Tool
using • 135
Multi Queue Message Tracker
properties • 235
MultiQueue Message Broadcaster
properties • 235
MultiQueue Message Filter
properties • 234
0
OSS Connectors
administrating • 187
running • 184
stopping • 184
ossadmin Administration Tool
using • 184
ossstart Administration Tool
using • 184
Outbound Email Queue
properties • 221
Outbound File System Queue
properties • 221
Outbound JMS Topic Queue
about • 222
properties • 222
Outbound Socket Queue
about • 223
properties • 223
P
P
Parameters
about • 245
Pause
about • 231
properties • 231
Processors

creating • 136	X
R	
Retry Handler about • 236 properties • 236	XML Parser about • 239 properties • 239 XML Validator
Roles	about • 230
configuring • 181	properties • 230
Routers Message Broadcaster • 234 Message Type Filter • 233 MultiQueue Message Broadcaster • 235 MultiQueue Message Filter • 234 MultiQueue Message Tracker • 235 Script Engine • 233	XSL Transmapper about • 240 properties • 240
S	
schemarefTool Administration Tool about • 173, 197 syntax • 199 using • 173, 174 Script Engine about • 233	
properties • 233	
specifying scripts • 140	
Scripts	
creating • 139	
Starting	
OSS Connector • 184	
Synchronizer Connector • 183	
Stopping OSS Connector a 184	
OSS Connector • 184	
Synchronizer Connector • 183 Synchronizer Connectors	
administrating • 185	
configuring • 185	
managing • 185	
modes • 183	
running • 183	
starting • 183	
stopping • 183, 192	
U	
Utility Processors	
DOM Builder • 229	
DOM Serializer • 229	
Encoding Converter • 229	
Message Counter • 230	
Pause • 231 XML Validator • 230	