SeeBeyond ICAN Suite

UCCnet ETD Library User's Guide

Release 5.0.5 for Schema Run-time Environment (SRE)



SeeBeyond Proprietary and Confidential

The information contained in this document is subject to change and is updated periodically to reflect changes to the applicable software. Although every effort has been made to ensure the accuracy of this document, SeeBeyond Technology Corporation (SeeBeyond) assumes no responsibility for any errors that may appear herein. The software described in this document is furnished under a License Agreement and may be used or copied only in accordance with the terms of such License Agreement. Printing, copying, or reproducing this document in any fashion is prohibited except in accordance with the License Agreement. The contents of this document are designated as being confidential and proprietary; are considered to be trade secrets of SeeBeyond; and may be used only in accordance with the License Agreement, as protected and enforceable by law. SeeBeyond assumes no responsibility for the use or reliability of its software on platforms that are not supported by SeeBeyond.

SeeBeyond, e*Gate, e*Way, and e*Xchange are the registered trademarks of SeeBeyond Technology Corporation in the United States and/or select foreign countries. The SeeBeyond logo, SeeBeyond Integrated Composite Application Network Suite, eGate, eWay, eInsight, eVision, eXchange, eView, eIndex, eTL, ePortal, eBAM, and e*Insight are trademarks of SeeBeyond Technology Corporation. The absence of a trademark from this list does not constitute a waiver of SeeBeyond Technology Corporation's intellectual property rights concerning that trademark. This document may contain references to other company, brand, and product names. These company, brand, and product names are used herein for identification purposes only and may be the trademarks of their respective owners.

© 2005 SeeBeyond Technology Corporation. All Rights Reserved. This work is protected as an unpublished work under the copyright laws.

This work is confidential and proprietary information of SeeBeyond and must be maintained in strict confidence.

Version April 5, 2005 8:25 pm.

Contents

Chapter 1	
Introduction	5
Intended Reader	5
UCCnet Overview	5
UCCnet Certification	7
HTTP(S) Interface UCCnet and HTTP: Overview	7 7
Supported Operating Systems	8
System Requirements	8
External System Requirements	8

Chapter 2

Installation	9
Installation on Windows Systems	9
ETD Library Installation Procedure	9
After Installation	10
UNIX Installation	10
ETD Library Installation Procedure	10
After Installation	10

Chapter 3

ETD LIbrary	11
UCCnet Messaging	11
UCCnet Message Structure	12
Message Elements	13
UCCnet Message Files	13
UCCnet Commands	15
UCCnet Documents	15
UCCnet ETD Library Files	16

Chapter 4

Implementation	18
Implementation Overview	18
Sample Schema	18
Schema Implementation UCCnet ETD Library Schema: Overview Schema Operation Schema Setup Schema Components	19 19 19 20 21
Creating the Sample Schema Creating the New Schema Creating Event Types and ETDs Creating Event Types Using the ETDs Creating and Configuring e*Ways Inbound and Outbound e*Ways Multi-Mode e*Ways Creating the e*Way Connections Configuring the IQ Manager Creating the IQs Creating Collaboration Rules Creating Collaborations Running the Schema	22 22 23 23 23 24 24 24 24 28 30 32 33 36 46 51

Index

52

Chapter 1

Introduction

This guide explains the SeeBeyond[™] Technology Corporation's (SeeBeyond[™]) UCCnet Event Type Definition (ETD) Library. This chapter provides an introduction to the guide and the library.

1.1 Intended Reader

The reader of this guide is presumed:

- To be a developer or system administrator with the responsibility for maintaining the e*Gate system
- To have expert-level knowledge of Windows operations and administration
- To be thoroughly familiar with Windows-style operations
- To have an understanding of the UCCnet system

UCCnet Overview

The UCCnet organization that created UCCnet is a wholly owned, not-for-profit subsidiary of the Uniform Code Council, Inc. (UCC). This organization is a virtual alliance of member companies in several industries engaged in the electronic exchange of information and services. Industry-leading companies are using the UCCnet to transform their industries by establishing common standards and practices.

The UCCnet system is used to communicate industry-compliant synchronized data throughout an industry's supply chain. Each data element of this industry-defined solution is clearly defined, including the processes of item introduction, change, and authorization. The UCCnet is a cost-effective means of transmitting information using standard Extensible Markup Language (XML) messaging schemas.

Note: The UCCnet does not replace the Electronic Data Interchange (EDI) standards, but it supports all the EDI data elements.

UCCnet provides the following services for member businesses:

- **Certification**: Transactions are certified, as well as dates and transaction levels (database of capabilities).
- **Business Data Pointers**: The UCCnet contains pointers to specialized business data via XML.
- **Base Addresses for Commerce**: There is a Domain Name System (DNS) entry for each supported transaction.
- **Subscription Options**: A business can subscribe at the corporate level by commerce type or business relationship, for example, a retailer subscribing to manufacturer promotions.

Once members are certified and incorporated into the UCCnet, they can access and use any of the provided services, as desired. The UCCnet provides a registry of core industry data including:

- Items
- Members

Also, as a service for its members, the UCCnet creates and supervises standardized messages for:

- UCCnet registry maintenance
- Commerce (between trading partners)
- Member interface processes

For certification and security purposes, the UCCnet has a database of capabilities that keeps track of the following information:

- Permissions to update the UCCnet registry
- Core information about business units, including commerce authority
- Data transmission security

For data transmission over its networks, the UCCnet uses a set of standardized XML message formats. The UCCnet ETD Library provides a corresponding set of Java programming language-based e*Gate ETD (.xsc) files.

ETD Library Components

The UCCnet ETD Library is made up of the individual .**xsc** files that comprise the library. See **"UCCnet ETD Library Files" on page 16** for a comprehensive list of the files.

Implementation Schema Sample

Installation of the UCCnet ETD Library includes a sample schema you can use to help you in implementing this library. See **Chapter 4** for details.

1.3 UCCnet Certification

This UCCnet ETD Library has been created and structured according to the certification guidelines provided in the UCCnet *Certification Program Guide for UCCnet Alliance Partners* (© 2001 UCCnet, Inc.; all rights are reserved). See this document for details on the UCCnet version 2.1 certification standards and process.

1.4 HTTP(S) Interface

The HTTP(S) interface for the UCCnet ETD Library and e*Gate is provided by the SeeBeyond HTTP(S) e*Way Intelligent Adapter. You must install this e*Way before you can use the library. For more information on how to use the e*Way, see the *HTTP(S) e*Way Intelligent Adapter User's Guide*.

1.4.1 UCCnet and HTTP: Overview

Communication between a trading partner and UCCnet using HTTP is based on the asynchronous *pull* communication model with respect to a trading partner. In this communication model, the trading partner always initiates the asynchronous communication.

HTTP is used to submit a particular request message from a trading partner to the UCCnet system and return the response message after UCCnet processes the message. The response message from UCCnet to a trading partner corresponding to a particular request message is retrieved by the trading partner in a different HTTP request-response than the HTTP request-response used to submit the request message.

Figure 1 shows a diagram of this model.



Figure 1 Request/response Communication Model

1.5 Supported Operating Systems

The UCCnet ETD Library is available on the following operating systems:

- Windows 2000, Windows XP, and Windows Server 2003
- HP Tru64 V5.1A
- HP-UX 11.0 and HP-UX 11i (PA-RISC)
- IBM AIX 5.1L and AIX 5.2
- Red Hat Linux Advanced Server 2.1 (Intel *x*86)
- Sun Solaris 8 and Solaris 9

1.6 System Requirements

To use the UCCnet ETD Library, you need to meet the following requirements:

- An e*Gate Participating Host
- A TCP/IP network connection
- A machine running Windows, to allow you to use the e*Gate Schema Manager and ETD Editor
- Java SDK version 1.3.1_02
- SeeBeyond HTTP(S) e*Way[™] Intelligent Adapter
- SeeBeyond XML Toolkit

The ETD Library and HTTP(S) e*Way must be administered using the Schema Manager.

1.7 External System Requirements

To use this ETD Library, you must

- Have access to the UCCnet system (version 2.1)
- Have an active UCCnet account
- Have a valid user name and password

For complete information on how to access and use this system, visit the following Web site:

http://www.uc-council.org/

Chapter 2

Installation

This chapter explains how to install the UCCnet Event Type Definition (ETD) Library.

2.1 Installation on Windows Systems

Pre-installation

- Exit all Windows programs before running the setup program, including any antivirus applications.
- You must have Administrator privileges to install this ETD library.

Note: See Chapter 3 for a list of the installed files that make up the ETD library.

2.1.1 ETD Library Installation Procedure

To install the UCCnet ETD Library on Windows systems

- 1 Log in as an Administrator on the workstation where you want to install the ETD library.
- 2 Insert the ETD library installation CD-ROM into the CD-ROM drive.
- ³ If the CD-ROM drive's **Auto-run** feature is enabled, the setup application should launch automatically; skip ahead to step 4. Otherwise, use the Windows Explorer or the Control Panel's **Add/Remove Applications** feature to launch the **setup.exe** file on the CD-ROM drive.
- 4 After the **InstallShield** setup application launches, follow the on-screen instructions to install the ETD library.

Be sure to install the ETD library files in the suggested **client** installation directory. The installation utility detects and suggests the appropriate installation directory.

Caution: Unless you are directed to do so by SeeBeyond support personnel, do not change the suggested installation directory setting.

2.1.2 After Installation

Once you have installed this ETD library, you must incorporate it into a schema by defining and associating the appropriate Collaborations, Collaboration Rules, Intelligent Queues (IQs), Event Types, and e*Ways before you can use the library to perform its intended functions.

2.2 UNIX Installation

Pre-installation

You do not require root privileges to install this ETD library. Log in under your name with which you wish to own the ETD library files. Be sure that this user has sufficient privileges to create files in the e*Gate directory tree.

2.2.1 ETD Library Installation Procedure

To install the UCCnet ETD Library on a UNIX system

- 1 Log in on the workstation containing the CD-ROM drive, and insert the CD-ROM into the drive.
- 2 If necessary, mount the CD-ROM drive.
- 3 At the shell prompt, type:
 - cd /cdrom/setup
- 4 Start the installation script by typing:

setup.sh

5 A menu of options appears. Select the **e*Gate Add-on Applications** option. Then, follow any additional on-screen directions.

Be sure to install the ETD library files in the suggested **client** installation directory. The installation utility detects and suggests the appropriate installation directory.

Caution: Unless you are directed to do so by SeeBeyond support personnel, do not change the suggested installation directory setting.

2.2.2 After Installation

Once you have installed this ETD library, you must incorporate it into a schema by defining and associating the appropriate Collaborations, Collaboration Rules, IQs, Event Types, and e*Ways before you can use the library to perform its intended functions.

Chapter 3

ETD Library

This chapter describes, and provides a reference for, the UCCnet Event Type Definition (ETD) Library, including a list of its component files.

3.1 UCCnet Messaging

The UCCnet defines a set of message conventions for the following purposes:

- To format its own messages
- To contain rules for carrying a UCCnet message within or on top of another protocol
- To process UCCnet messages along the UCCnet message path

Complete UCCnet specifications can be found at the Uniform Code Council (UCC) Web site as follows:

http://www.uc-council.org/

The UCCnet ETD Library follows the UCCnet standards and conventions, allowing you to use them to transmit and process, through e*Gate Integrator, any UCCnet message data.

Note: The UCCnet ETD Library is compatible with UCCnet version 2.1.

3.1.1 UCCnet Message Structure

Figure 2 shows a diagram of a UCCnet message and its components.



UCCNet Envelope	
UCCNet Header	
XML Information	
UCCNet	
Body	
XMLMessage	
Includes:	
Response or	
Transaction	
Entity	
Identification	
Commands	

UCCnet messages consist of the following major parts:

- A UCCnet envelope that marks the start and end of the UCCnet message, as well as defining a framework for describing what is in a message and how to process it
- A UCCnet header encoded in the Extensible Markup Language (XML), which carries general information about the UCCnet message
- A UCCnet body that carries the actual message payload
- *Note:* For complete information on UCCnet messaging, see the UCCnet **Technical User's** *Guide*.

3.1.2 Message Elements

The UCCnet message envelope contains the header and body, and the body in turn contains *either* a response or a transaction. Each transaction contains an identification and one or more commands.

UCCnet messages can have any of the following basic elements:

- **envelope:** Contains the header and message body.
- messageHeader: An XML header.
- **body:** Contains the message and can be either a response or a transaction.
- **Response:** A response to an earlier message.
- **transaction:** Contains an identification and a command.
- entityIdentification: An identification for the message.
- **command:** Can be one or more of the following commands:
 - CheckComplianceCommand
 - DocumentCommand
 - DocumentIdentificationCommand
 - NotificationStateCommand
 - QueryCommand
 - LinkCommand
 - PublishCommand

Note: See "UCCnet Message Files" on page 13 for an explanation of the commands.

Of all the elements shown in the previous list, only **response** and the seven commands are valid UCCnet messages, and there is a corresponding ETD file for each in the library. If your message contains more than one command, you must use the **UCCnet21.xsc** ETD file, which contains all the commands (see **Table 1 on page 17**).

3.1.3 UCCnet Message Files

The UCCnet message files can be either a response or a command. The files are:

- **Response**: Contains a response to a previously sent message.
- **CheckComplianceCommand**: Allows a trading partner to check the validity of a document or command before actually sending the command anywhere to be executed.

- **DocumentCommand**: Used to update the status of notifications contained in a user's work list. This command is also used to provide information to other users and to possibly fit into a normal cycle of work-list maintenance. It requires that one of the following actions are taken:
 - ADD: Instructs the receiving application to store documents.
 - CHANGE_BY_REFRESH: Instructs the receiving application to update existing documents. This action is meant to update documents by total replacement. This means that users must transmit complete images of documents to be changed, and not simply the elements to be changed.
 - **INITIAL_LOAD**: A type of publication designed to either assist a new supply side Trading Partner to initially publish their items to existing UCCnet demand side users, or to assist new demand side users in requesting publication of information from existing supply side UCCnet participants.
 - **CORRECT**: Instructs the receiving application to correct the specified document. When the document is an item, this command does not create a new version of the item. Users must transmit complete images of the documents to be corrected.
- **DocumentIdentificationCommand**: Can only be used with documents according to the UCCnet specifications (see the UCCnet documentation for details) and requires the following information:
 - **DocumentIdentificationCommandType**: Type of command to execute, that is, DELETE, WITHDRAW, DELIST, REQUEST_PUBLICATION, and REQUEST_INITIAL_LOAD_PUBLICATION.
 - **entityIdentification**: Command identifier to be used in acknowledgement response.
 - documentIdentificationList: List of document identifiers to be affected.
- NotificationStateCommand: Used to update the status of notification contained in a user's work list. This command is also used to provide information to other users, and to possibly fit into a normal cycle of work-list maintenance. The command is set to following values:
 - **UNREAD**: Notification has not been read. This is the default state when the notification is first inserted into the work list.
 - **READ**: Notification has been read via the user interface or through XML. This state is applied to all notifications automatically.
 - ACKNOWLEDGED: Notification state used to convey information.
 - **PROCESSED**: Notification state used to convey that the contents have been processed.
 - **DELETE**: Remove the notification from the work list.

- **QueryCommand**: Used to pull information from the application. Types of documents that can be pulled include, for example, notifications, items, and parties. A query command returns an acknowledgement that contains a results list with the appropriately selected results.
- LinkCommand: Establishes parent-child or peer relationships between two or more documents. When this command is received, a link between the two entities is created. This link is visible the next time the parent or the first peer is queried or viewed.
- **PublishCommand**: Publishes (sends) information to other trading partners utilizing UCCnet. This command can publish content registered with the UCCnet service (item or party). It can publish documents stored externally to the system (price or bracket). It can also be used to publish acknowledgements of receipt and be processed after a document is received by a trading partner.

The UCCnet ETD Library contains ETDs that correspond to and enable each of these message types and/or commands.

UCCnet Commands

UCCnet commands are used to send a request to the UCCnet service, to perform a task. All commands are transient and only last until the execution of the command.

Each command defines an action to be done by the receiving system. Sometimes the action is defined by the command itself. At other times, the action is defined as an attribute of the command. A command can consist of any one of the command types, which are also message types, as explained under "UCCnet Message Files" on page 13.

UCCnet Documents

UCCnet employs 29 documents as part of its messaging. These documents are contained in the **UCCnet21.xsc** file of the ETD Library (see **Table 1 on page 17**). The document names are:

- Access Point
- Acknowledgement
- Allowance Or Charge
- Authorization
- Authorization Item Group
- Authorization Document
- Bracket List
- Capability

- Category
- Certificate
- Filter
- Item
- Item Group
- Market Group
- Notification
- Party
- Payment Term
- Price
- Pricing Lead Time
- Publication
- Publication Receipt Notification
- Query Document
- Role
- Subscription
- Trading Partner
- User
- User Authorization Permission Group
- Extended Data Definition
- Extended Data
- *Note:* For complete information on these documents, see the UCCnet *Technical User's Guide*.

3.2 UCCnet ETD Library Files

All the files listed in **Table 1 on page 17** are installed in the **eGate\server\registry\repository\default\templates\UCCnet\UCCnet21** directory, where **eGate** is the e*Gate installation directory. In addition, there is a .jar file installed for each .xsc file.

Table 1 lists the .**xsc** files contained in the UCCnet ETD Library, along with a description of the file and/or the file's corresponding UCCnet message type.

ETD File Name	Description and/or UCCnet Message Type
UCCnet21.xsc	Container for entire library, including commands and documents.
Response_E.xsc	Corresponds to response message type only.
CheckComplianceCommand_E.xsc	Corresponds to CheckComplianceCommand message type and envelope.
LinkCommand_E.xsc	Corresponds to DocumentCommand message type and envelope.
DocumentIdentificationCommand_E.xsc	Corresponds to DocumentIdentificationCommand message type and envelope.
NotificationStateCommand_E.xsc	Corresponds to NotificationStateCommand message type and envelope.
QueryCommand_E.xsc	Corresponds to QueryCommand message type and envelope.
LinkCommand_E.xsc	Corresponds to LinkCommand message type and envelope.
PublishCommand_E.xsc	Corresponds to PublishCommand message type and envelope.

Table 1 UCCnet ETD Library Files

Chapter 4

Implementation

This chapter explains how to implement the UCCnet Event Type Definition (ETD) Library, using a sample schema.

4.1 Implementation Overview

This section explains how to use the UCCnet ETD Library, including an e*Gate Integrator sample schema included on your installation CD-ROM. Find this sample on the CD-ROM at the following path location:

\samples\uccnet

This sample allows you to observe and/or create an end-to-end data-exchange scenario involving e*Gate, the library files, and the HTTP(S) e*Way Intelligent Adapter. This chapter explains how to implement this sample schema that uses the UCCnet ETD Library.

4.2 Sample Schema

You can use the sample schema to help you learn how to implement the UCCnet ETD Library. To use this schema, you must first import it into e*Gate.

Caution: A control block .*dtd* file is required in order to use UCCnet. You can download this file from the UCCnet Web site. This file is *not* contained in the ETD Library or sample schema. You can also create your own control block file, if desired. Use the XML Toolkit's DTD Wizard to convert the .*dtd* file to an .*xsc* file, which must be accessible to e*Gate. The UCCnet Web site is:

http://www.uc-council.org/

Before Importing the Sample Schema

To import the schema, the UCCnet ETD Library must be installed, as well as the HTTP(S) e*Way. All of the necessary files and scripts must be located in the default location, and you must also have access to the UCCnet.

Import the sample schema as follows:

- 1 Copy the desired .**zip** file from the **\Samples\uccnet** directory in the install CD-ROM to your desktop or to a temporary directory, then unzip the file.
- 2 Start the e*Gate Schema Manager.
- 3 On the **Open Schema from Registry Host** dialog box, click **New**.
- 4 On the New Schema dialog box, click Create from export, and then click Find.
- 5 On the **Import from File** dialog box, browse to the directory that contains the sample schema.
- 6 Click the .zip file then click Open.

The schema is installed.

4.3 Schema Implementation

This section explains how to implement a sample schema for the UCCnet ETD Library. The schema demonstrates how to configure the essential features of the library in a typical e*Gate environment, including connectivity with the SeeBeyond HTTP(S) e*Way Intelligent Adapter.

4.3.1 UCCnet ETD Library Schema: Overview

This section provides an overview of how to configure the sample schema and how it operates. The name of this schema is UCCnetSampleSchema, and it is contained in the import file **UCCnetSampleSchema.zip**.

Schema Operation

This sample schema has the following input/output setup:

- **Input**: Request HTTP message via UCCnet.
- **Output**: Response HTTP message via UCCnet.

Each of the major steps in schema operation happens via an e*Way component. The schema passes data through e*Gate using ETDs based on those found in the UCCnet ETD Library.

The UCCnetSampleSchema processes the request and reply messages using the following general steps:

- The schema gets a UCCnet HTTP *request* message from an external system.
- The schema processes the request message as follows:
 - Attaches a unique message identifier
 - Creates a control block
 - Generates an HTTP request message
 - Sends the HTTP request message to UCCnet via the HTTP(S) e*Way
- The schema picks up the control block and sends it to the UCCnet response handle, to get the *response* message in return.
- The schema sends the HTTP response message to the desired location.
- *Note:* For complete information on UCCnet and its messaging protocols, see the appropriate UCCnet documentation.

Schema Setup

Figure 3 on page 21 shows the schema's general architecture, using the e*Gate Schema Manager's Network View.





Schema Components

This sample schema implementation consists of the following basic components:

- **ewGetRequestMessage**: Inbound file e*Way that brings the UCCnet request message into e*Gate.
- **mmGenerateHTTPRequest**: HTTP(S) e*Way (Multi-Mode) that processes the request message, including adding an identifier and creating a control block; the e*Way then sends the processed request message to the UCCnet.
- **mmGetHTTPResponse**: HTTP(S) e*Way (Multi-Mode) that picks up the control block and sends it to the UCCnet response handle, to get the response message in return.
- **ewPutResponseMessage**: Outbound file e*Way that sends the HTTP response message to the desired location.
- colGetRequestMessage, colGenerateHTTPRequest, colGetHTTPResponse, and colPutResponseMessage: Collaborations for their respective e*Ways.

- **iqUCCnet**, **iqControlBlock**, and **iqResponse**: Intelligent Queues (IQs) that queue data as it is sent through the e*Gate system.
- **localhost_iqmgr**: SeeBeyond Java Messaging Service (JMS) IQ Manager that manages the IQs.
- **ecToUCCnet**: e*Way Connection for the **GenerateHTTPRequest** e*Way.
- **ecFromUCCnet**: e*Way Connection for the **GetHTTPResponse** e*Way.

4.4 Creating the Sample Schema

This section explains the basic steps for how to create the sample schema UCCnetSampleSchema.

4.4.1 Creating the New Schema

The first task in deploying the schema example is to create a new schema name. While it is possible to use the default schema for this example implementation, it is recommended that you create a separate schema for testing purposes.

After you install the HTTP(S) e*Way, do the following steps:

- 1 Start the e*Gate Schema Manager.
- 2 When the Schema Manager prompts you to log in, select the host that you specified during installation, and enter your password.
- 3 You are then be prompted to select a schema. Click on **New**.
- 4 Enter a name for the new Schema. In this case, enter **UCCnetSampleSchema**, or any name as desired.

The Schema Manager opens under your new schema. You can access the ETD Editor and Collaboration Rules Editor features via the Schema Manager. You are now ready to begin creating the necessary components for this sample schema.

Figure 4 on page 23 shows an example of the Schema Manager window for this schema.

Note: For complete information on how to set up an e*Gate schema, see the e*Gate Integrator User's Guide and Creating and End-to-end Scenario with e*Gate Integrator.



Figure 4 Schema Manager Main Window for UCCnetSampleSchema

4.4.2 Creating Event Types and ETDs

The HTTP(S) e*Way installation includes the file **httpclient.xsc**, which represents an HTTP(S) ETD template.

Creating Event Types

Using the Schema Manager, create the following Event Types:

- HTTPClient
- UCCnet

Using the ETDs

In this sample schema, you use the **httpclient.xsc** ETD file and installed UCCnet ETD Library files.

Note: You must create a control block .*xsc* file by importing the control block from UCCnet. Through the use of the XML Toolkit, use this file to create and compile an *xsc* file.

4.4.3 Creating and Configuring e*Ways

The first components to be created are the following e*Ways:

- Inbound: ewGetRequestMessage
- Outbound: **ewPutResponseMessage**
- Multi-Mode e*Ways:
 - mmGenerateHTTPRequest

mmGetHTTPResponse

This section provides instructions for creating the e*Ways.

Inbound and Outbound e*Ways

To create the e*Ways

- 1 Select the e*Gate Schema Manager's **Components** tab.
- 2 Open the host on which you want to create the e*Ways.
- ³ Select the **Control Broker** that manages the new e*Ways.
- 4 On the palette, click the e*Way icon.
- 5 Enter the name of the new e*Way (in this case, **ewGetRequestMessage)**, then click **OK**.
- 6 Select ewGetRequestMessage, then double-click to edit its properties.
- 7 When the e*Way Properties dialog box appears, click on the Find button beneath the Executable File field, and select stcewfile.exe for the executable file (see Figure 5 on page 25).

🌐 e*Way - ewGetRequestMessage Prop	perties	_ 🗆 ×
General Start Up Advanced Security		
ewGetRequestMessage		
Executable file		
bin\stcewfile.exe		
	Clear	Find
Additional command line arguments:		
% -un %_USERNAME% -up %_PASSWOR	:D% -rp %_	REGPORT%
Run as user		
Administrator		•
Configuration file		
configs\stcewfile\ewGetRequestMessage.cfg		
Clear	Find	Edit
OK Cancel A	Apply	Help

Figure 5 ewGetRequestMessage e*Way Properties Dialog Box

Configure the e*Way properties as shown in the previous figure.

8 Under the **Configuration File** text box, click on the **New** button. When the **Settings** configurations of the e*Way Editor appear, set the parameters for **Settings** as shown in Table 2.

Table 2 Configuration Parameters for ewGetRequestMessage e*Way

Parameter	Value
General Settings	
AllowIncoming	Yes
AllowOutgoing	No
Outbound Settings	Default
Poller Inbound Settings	
PollDirectory	C:\Indata (input file folder)
InputFileExtension	*.fin (input file extension)
PollMilliseconds	Default

Parameter	Value
Remove EOL	Default
MultipleRecordsPerFile	Default
MaxBytesPerLine	Default
BytesPerLineIsFixed	Default

Table 2 Configuration Parameters for ewGetRequestMessage e*Way (Continued)

Figure 6 shows an example of the e*Way Editor Window.

Figure 6 e*Way Editor Window for ewGetRequestMessage

Edit Settings for C:/EGATE/Client/configs//ewGetReque File View Options Help	estMessage.cfg
Goto Section: General Settings	- <u>•</u> • •
Goto Parameter: AllowIncoming	_
AllowIncoming	
O NO O YES	
AllowOutgoing	
NO YES	
PerformanceTesting	<u>ra 20 (11)</u>
NO YES	

- 9 Save the **.cfg** file, and exit from **Settings**.
- 10 After selecting the desired parameters, save the configuration file and promote the file to run time. Close e*Way Editor and the **.cfg** file.

- 11 Use the **Startup**, **Advanced**, and **Security** tabs to modify the default settings for each e*Way you configure as follows:
 - A Use the **Startup** tab to specify whether the e*Way starts automatically, or restarts after abnormal termination or due to scheduling, and so on.
 - **B** Use the **Advanced** tab to specify or view the activity and error logging levels, as well as the Event threshold information.
 - C Use **Security** to view or set privilege assignments.
- 12 Click **OK** to close the **e*Way Properties** dialog box.
- 13 Repeat steps 4 through 10 for the **ewPutResponseMessage** e*Way (see Figure 7).

Figure 7 ewPutResponseMessage e*Way Properties Dialog Box

🌐 e*Way - ewPutResponseMessage Properties	_ 🗆 🗙
General Start Up Advanced Security	
ewPutResponseMessage	
Executable file	
bin\stcewfile.exe	
Clear	Find
Additional command line arguments:	
% -un %_USERNAME% -up %_PASSWORD% -rp %	_REGPORT%
Run as user	
Administrator	•
- Configuration file	
Configuration file	
configs\stcewfile\ewPutResponseMessage.cfg	
configs\stcewfile\ewPutResponseMessage.cfg Clear Find	Edit
configs\stcewfile\ewPutResponseMessage.cfg Clear Find	Edit
configs\stcewfile\ewPutResponseMessage.cfg Clear Find	Edit

Configure the e*Way properties as shown in the previous figure. Use the configuration parameters shown in **Table 3 on page 28**.

Parameter	Value
General Settings	
AllowIncoming	No
AllowOutgoing	Yes
Outbound Settings	
OutputDirectory	C:\DATA\HTTP
OutputFileName	output%d.dat
MultipleRecordsPerFile	Yes
MaxRecordsPerFile	10000
AddEOL	Yes
Poller Inbound Settings	Default
Performance Testing	Default

Table 3 Configuration Parameters for ewPutResponseMessage e*	Nay
--	-----

14 Repeat step 11 for the new e*Way. When you are finished click **OK**.

Multi-Mode e*Ways

To create the Multi-Mode e*Ways

- 1 Select the e*Gate Schema Manager's **Components** tab.
- 2 Open the host on which you want to create the e*Way.
- ³ Select the Control Broker that manages the new e*Way.
- 4 On the palette, click the icon to create a new e*Way.
- 5 Enter the name of the new e*Way (**mmGenerateHTTPRequest**), then click **OK**.
- 6 Select the new component, then double-click to edit its properties.
- 7 When the e*Way Properties dialog box appears, use the default executable file, stceway.exe (see Figure 8 on page 29).

🌐 e*Way - mmGenerateHTTPRequest Properties 👘 💶 💌
General Start Up Advanced Security
mmGenerateHTTPRequest
Executable file
bin\stceway.exe
Clear Find
Additional command line arguments:
% -un %_USERNAME% -up %_PASSWORD% -rp %_REGPORT%
Run as user
Administrator
Configuration file
configs\stceway\mmGenerateHTTPRequest.cfg
Clear Find Edit
OK Cancel Apply Help

Configure the e*Way properties as shown in the previous figure.

- 8 To edit the JVM Settings, select **New** under the **Configuration File** text box. Use the default configuration parameters.
- *Note:* See the *HTTP(S) e***Way Intelligent Adapter User's Guide* for details on the parameters associated with the Multi-Mode e*Way for the HTTP(S) e*Way.
 - 9 Save the .cfg file, and exit from Settings.
 - **10** Use the **Startup**, **Advanced**, and **Security** tabs to modify the default settings for each.
 - A Use the **Startup** tab to specify whether the Multi-Mode e*Way starts automatically, restarts after abnormal termination or due to scheduling, etc.
 - **B** Use the **Advanced** tab to specify or view the activity and error logging levels, as well as the Event threshold information.
 - C Use **Security** to view or set privilege assignments.
 - 11 Select **OK** to close the **e*Way Properties** dialog box.

12 Repeat steps 4 through 11 for the **mmGetHTTPResponse** e*Way (see Figure 9).

Figure 9 mmGetHTTPResponse e*Way Properties Dialog Box

e*Way - mmGetHTTPResponse Properties
General Start Up Advanced Security
mmGetHTTPResponse
Executable file
bin\stceway.exe
Clear Find
% -un %_USERNAME% -up %_PASSWORD% -rp %_REGPORT%
Administrator
Configuration file
configs\stceway\mmGetHTTPResponse.cfg
Clear Find Edit
OK Cancel <u>A</u> pply Help

Configure the e*Way properties as shown in the previous figure. Use the default configuration parameters.

4.4.4 Creating the e*Way Connections

The e*Way Connection configuration file contains the connection information, along with the information needed to communicate via HTTP(S).

To create and configure new e*Way Connections

- 1 Highlight the **e*Way Connection** folder on the **Components** tab of the e*Gate Schema Manager.
- 2 On the palette, click the icon to create a new e*Way Connection.
- 3 Enter the name of the e*Way Connection (ecToUCCnet), then click OK.
- 4 Select the new **e*Way Connection**, then right-click to edit its properties.

5 When the **Properties** dialog box opens, select **HTTP/HTTP(S)** from the **e*Way Connection Type** drop-down menu (see Figure 10).

Figure 10 ecToUCCnet e*Way Connection Properties Dialog Box

erway connection - ecrooccheci	Properties
General	
e*Way Connection Type: HTTP(S)	V
Event Type "get" interval	
Time (in milliseconds) after a retrieva event available" before attempting an	al returns "no hother
retrieval.	
retrieval.	10000
retrieval.	10000
e*Way Connection Configuration File	10000
retrieval. e*Way Connection Configuration File- configs\httpclient\ecToUCCnet.cfg Clear	10000 r Find Edit
retrieval. e*Way Connection Configuration File- configs\httpclient\ecToUCCnet.cfg Clear	10000 r Find Edit
retrieval. e*Way Connection Configuration File- configs/httpclient/ecToUCCnet.cfg Clear	10000 r Find Edit

Configure the e*Way Connection properties as shown in the previous figure.

- 6 Under e*Way Connection Configuration File, click New.
- 7 After the e*Way Editor opens, select the desired parameters.
- *Note:* For more information on the e*Way's *Connection Type* parameters, see the *HTTP(S)* e*Way Intelligent Adapter User's Guide.
 - 8 Save the .cfg file and promote it to run time.

9 Repeat steps 2 through 8 for the ecFromUCCNet e*Way Connection (see Figure 11).

Figure 11 ecFromUCCnet e*Way Connection Properties Dialog Box

e*Way Connection - ecFromUCCnet Properties	_ 🗆 🗙
General	
e*Way Connection Type: HTTP(S)	-
Event Type "get" interval	
Time (in milliseconds) after a retrieval returns "no event available" before attempting another	
retrieval.	
retrieval.	10000
retrieval.	10000
e*Way Connection Configuration File	10000
e*Way Connection Configuration File configs\httpclient\ecFromUCCnet.cfg	10000 Edit
retrieval. e*Way Connection Configuration File configs%httpclient%cFromUCCnet.cfg Clear Find	10000 Edit

Configure the e*Way Connection properties as shown in the previous figure.

4.4.5 Configuring the IQ Manager

You must configure the IQ Manager in your schema to use the SeeBeyond Java Messaging Service (JMS) IQ Service. The system has already named the IQ Manager **localhost_iqmgr** for you.

IQs use IQ Services to transport data. IQ Services provide the mechanism for moving Events between IQs, handling the low-level implementation of data exchange (such as system calls to initialize or reorganize a database).

To configure the IQ Manager

- 1 Select the e*Gate Schema Manager's **Components** tab.
- 2 Open the host for the IQ Manager.
- 3 Select the Control Broker that manages the IQ Manager.
- 4 Double-click on the **IQ Manager** icon.

5 For the IQ Manager Type, select SeeBeyond JMS (see Figure 12).

Figure 12 localhost_iqmgr IQ Manager Properties Dialog Box

🜐 IQ Manager - localhost_iqmgr Properties
General Start Up Advanced Security
localhost_iqmgr
IQ Manager Type
SeeBeyond JMS
Additional command line arguments:
E% -un %_USERNAME% -up %_PASSWORD% -rp %_REGPORT%
Run as user
Administrator
Configuration file
configs\stcmsagentVocalhost_iqmgr.cfg
Clear Find Edit
OK Cancel <u>A</u> pply Help

Configure the IQ Manager properties as shown in the previous figure.

6 Click **OK**.

4.4.6 Creating the IQs

The next step is to create and associate the schema's IQs. These components manage the exchange of information within the e*Gate system, providing non-volatile storage for data as it passes from one component to another.

To create the IQs

- 1 Select the Navigation pane's **Components** tab.
- 2 Open the host on which you want to create the IQ.
- 3 Open a Control Broker.
- 4 Select the **IQ Manager** icon.
- 5 On the palette, click the **IQ** icon.

- 6 Enter the name of the new IQ (iqUCCnet), then click OK.
- 7 Select the new IQ, then double-click to edit its properties.
 - The **IQ Properties** dialog box appears.
- 8 On the **General** tab, specify the **Service** and the **Event Type Get Interval**.

Use the default **Event Type Get Interval** of 100 ms for this implementation.

- 9 On the **Advanced** tab, be sure that **Simple publish/subscribe** is checked under the **IQ Behavior** section.
- 10 Close the **IQ Properties** dialog box.
- 11 Repeat steps 5 through 10 for the **iqControlBlock** and **iqResponse** IQs.

Figure 13 through **Figure 15 on page 36** show the **IQ Properties** dialog boxes for all three IQs.

🖥 IQ - iqUCCnet Prop	erties		
General Database E	xternal Adva	nced	
iqUCCnet			
Service:	C_JMS_JQ		<u>.</u>
Initialization string:			
Event Type "get" inte	rval		
Time (in millisecond: empty	s) to wait whe	n the IQ is	
			100
Auto Recovery			

Figure 13 iqUCCnet IQ Properties Dialog Box

🌐 IQ - iqControlBlock Properties 📃 🚺 🔪
General Database External Advanced
iqControlBlock
Service: STC_JMS_IQ
Initialization string:
Event Type "get" interval Time (in milliseconds) to wait when the IQ is empty before performing another "get" operation: 100
Auto Recovery
OK Cancel <u>A</u> pply Help

Figure 14 iqControlBlock IQ Properties Dialog Box

🜐 IQ - iqResponse Properties
General Database External Advanced
iqResponse
Service: STC_JMS_IQ
Initialization string:
Event Type "get" interval Time (in milliseconds) to wait when the IQ is empty before performing another "get" operation:
100
Auto Recovery
OK Cancel <u>A</u> pply Help

Figure 15 iqResponse IQ Properties Dialog Box

Configure the IQ properties as shown in the previous figures. Be sure all the IQ **Start Up** tab settings are set to **Start automatically**.

4.4.7 Creating Collaboration Rules

The next step is to create the Collaboration Rules that extract and process selected information from the source Event Type defined previously, according to its associated Collaboration Service. The **Default Editor** can be set to either the Monk or Java programming language.

From the **Schema Manager Task Bar**, select **Options** and click **Default Editor**. For this example, set the default to Java.

To create a Collaboration Rules file

- 1 Select the **Components** tab in the e*Gate Schema Manager.
- 2 In the Navigation pane, select the **Collaboration Rules** folder.
- 3 On the palette, click the **Collaboration Rules** icon.
- 4 Enter the name of the new Collaboration Rule, then click **OK**. **crGenerateHTTPRequest** is used for this example.



- 5 Select the new Collaboration Rule, then right-click to edit its properties.
- 6 The **Collaboration Rules crGenerateHTTPRequest Properties** dialog box appears (see Figure 16).

🌐 Collaboration Rules - crGenerateHTTPRequest Properties 📃 🔲 🗙
General Subscriptions Publications Collaboration Mapping
Service: Java
Collaboration Rules
collaboration_rules\crGenerateHTTPRequest.class
Clear Find Edit Test
Initialization file
collaboration_rules\crGenerateHTTPRequest.ctl
Clear Find
OK Cancel <u>A</u> pply Help

Figure 16 crGenerateHTTPRequest Properties Dialog Box: General Tab

On the **General** tab in the dialog box select the **Java Collaboration Service**. The Collaboration Rules use the e*Gate Java Collaboration Service to manipulate Events or Event data.

- 7 In the **Initialization String** text box, enter any required initialization string that the Collaboration Service may require. This field can be left blank.
- 8 Click the **Collaboration Mapping** tab.

9 Using the **Add Instance** button, create instances to coincide with the Event Types (see Figure 17).

Figure 17 crGenerateHTTPRequest Properties: Collaboration Mapping Tab

Instance Name	ETD		Mode	Trigger	Manual Publish
http	httpclient.xsc	Find	Out	N/A	
request	UCCnet21.xsc	Find	In		N/A
output	GenericOutEvent.xsc	Find	Out	N/A	

Configure the rest of the Collaboration Rule as shown in the previous figure.

- *Note:* The input is the control block .*xsc* file you create by importing the control block from UCCnet. Through the use of the XML Toolkit, use this file to create and compile an *xsc* file.
 - 10 Select the **General** tab again, then click **New**.

The Collaboration Rules Editor Main window opens.

11 Expand the window to full size for optimum viewing, expanding the source and destination Events also (see Figure 18).

Figure 18 Collaboration Rules Editor: Generate Unique Request Message Rule



The previous figure shows the properties of the Generate Unique Request Message Identifier Rule properties. Figure 19 on page 40 shows the Set Control Block Related Data Rule properties.

다. Collaboration Roles Editor - urGenerateHTTPRequest (Sandbox - M	dified)
244	
To the Types	Perstinution Events
≘.=tre.uest[E v=l.µ=]	ստերտ[]Serieric3.tEve t[] ¶‡ ձեր [HupCle, t] "¶ ⊡
C blocs → nethod ♥ var L for v. f (① trz i setch i opp i deterve i deterve i suaness des = ** Generatel ITPRequest ⊕ -% of Generatel IT-Request ⊕ -% of Generatel IT-Request ⊕ -% of Generatel IT-Request	rule E. switch care Li while Li do 🔷 return 1 throw weetwarp 1 anyoin Rule Properties Description: po: Lottro Bod-rokted data
definition of the second data definition of the second data	Rule:
	Documentation:

Figure 19 Collaboration Rules Editor: Set Control Block Related Data Rule

Note: Be sure to configure the control block code for this Collaboration Rule. Also, be sure to include the user name and password.

Figure 20 on page 41 shows the set http request data Rule properties.

🕞 Collaboration Rules Editor - crGenerateHTTPRequest ((Sandbox - Modified)									_ 8	×
File Edit View Tools Help											
ana Ti Source Events				-	Dectir	nation Eve	anto				
⊕ •t request [Envelope]								output [G	http [HttpC	Client] 📲	
O block Image: method Var Image: method for Law (1) try ! catch (1) copy (1) datamap (1)	if () rule	E switch	→ cas	e 🕒	while	Ľ	do 🥏	return	! thro	w	
Business Rules											
GreenerateHTTPRequest GreenerateHTTPRequest GreenerateHTTPRequest GreenerateHTTPRequest GreenerateHTTPRequest GreenerateUnique Request Message Identifier O set ControlBlock related data O set ControlBlock and send it to iqControlBlock GreenerateUnique Greenerat		Rule P Descrip Rule: String m gethttp(gethttp(roperties ion: set http: ig = STCType).setTextData i.post();	Converte	data er.toStrinç	j(getrequ	est().marsh	al());			4
		Docume	ntation:							V	
											-

Figure 20 Collaboration Rules Editor: http request data Rule

12 You must create the Collaboration Rules class.

Note: See the e*Gate Integrator User's Guide for details on this procedure.

You can create this class as desired or use the previous figures as a model.

- 13 Before compiling the code, on the **Tools** menu, click **Options**.
- 14 Verify that all necessary **.jar** files are included. For clear HTTP (SSL not enabled), ensure that the **stchttp.jar** and **stcutils.jar** file are added.
- 15 For a Collaboration that uses SSL, ensure that the **jcert.jar**, **jnet.jar**, and **jsse.jar** files are included.
- 16 When you have finished defining all the desired business logic, compile the Java code by selecting **Compile** from the **File** menu.

If the code compiles successfully, the message **Compile Completed** appears. If the outcome is unsuccessful, a Java Compiler error message appears. If there are any Java errors, be sure to correct them.

The **Save** menu opens.

- 17 Provide a name for the **.xpr** file. For this example, use **crGenerateHTTPRequest.xpr**.
- 18 Once the compilation is complete, save the Collaboration Rules file and exit the Collaboration Rules Editor.
- 19 Repeat steps 3 through 18 to create the crGetHTTPResponse Collaboration Rule. See Figure 21 through Figure 24 on page 45 for the Collaboration Rule properties details, or you can create your own Business Rules.

Figure 21 crGetHTTPResponse Properties Dialog Box: General Tab

🌐 Collaboration Rules - crGetHTTPResponse Properties
General Subscriptions Publications Collaboration Mapping
Service: Java
Initialization string:
Collaboration Rules
collaboration_rules\crGetHTTPResponse.class
Clear Find Edit Test
Initialization file
collaboration_rules\crGetHTTPResponse.ctl
Clear Find
OK Cancel Apply Help

eneral Subscripti	ons Publications Collaborat	tion Mapping			
Instance Name	ETD		Mode	Trigger	Manual Publish
http	httpclient.xsc	Find	Out	N/A	
input	GenericOutEvent.xsc	Find	In		N/A
response	UCCnet21.xsc	Find	Out	N/A	
		Add In	stance	Re	move Instance

Figure 22 crGetHTTPResponse Properties: Collaboration Mapping Tab

File Edit Wew Tools Help Source Events if " 2 input [GenericOutEvent] U block method var method var	Scollaboration Rules Editor - crGetHTTPResponse (Sandbox - Modified)	
Source Events Source Events Texponse [Envelope] Controllevent] Texponse [Envelope] Controllevent] Texponse [Envelope] Controllevent] Texponse [Envelope] Controllevent] Controllevent] Texponse [Envelope] Controllevent] Cont	File Edit View Tools Help	
Construction of the second of the secon	PR	PIP Destination Supply
() block ● try try ist for ist	⊞ "tinput [GenericOutEvent]	response [Envelope] •
① try 1 catch 0 copy ① list lookup ① timestamp ① uniqueid Business Rules GetHITPResponse executeBusinessRules fetBoolean O fetBoolean fetBoolea	O block 🖘 method 🖉 var 🖿 for 🕹 if O rule	E, switch → case 🗅 while 🗠 do 🛹 return I throw
Business Rules Rule Properties	() try ! catch () copy () datamap () list lookup () timestamp	() uniqueid
Rule Properties Rule Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http://ient TextData to be the input (ControlBlock) and get UCCnet response Image: Set Http:/	Business Rules	
<pre>content in the spunder content in the sp</pre>	□	Rule Properties
<pre>vector() set Http://within.com/vector() and get UCCnet response</pre>	Brews creatin i PResponse Frees executeBusinessRules	Descriptions (at Http://apt.taytData.to.be.the.input (ControlPlack) and act USCont response
<pre> Click errors and unmarshal the response O Check errors and unmarshal the response return userInitialize userTerminate UserTerm</pre>	🖉 🗸 🖉 retBoolean	Description; per ricipalent rexubate to be the input (contrololock) and get occhet respon
C) Check errors and unmarshal the response. String Text_Result; do 4 UserInitialize wiserTerminate do 4 Thread.sleep(2000); Text_Result(),getPayload()); gethttp().getHttpResult().getTextResult(); noResponse Text_Result.indexOf("html"); while (noResponse >=0) Documentation:	$-\!-\!()$ set HttpClient TextData to be the input (ControlBlock) and get UCCnet response	Kule:
<pre>double control co</pre>	Check errors and unmarshal the response	String Text_Result;
<pre> userTerminate userTerminate userTerminate userTerminate Thread.sleep(2000); gethttp().setTextData(getinput().getPayload()); gethttp().post(); Text_Result_getinput().getTextResult(); noResponse = Text_Result.indexOf("html"); while (noResponse >=0) Documentation: </pre>	·····≪∞ return •·····S userInitialize	do /
Documentation:	Sucramonate userTerminate	<pre>UU 1 Thread.sleep(2000); gethttp().setTextData(getinput().getPayload()); gethttp().post(); Text_Result = gethttp().getHttpResult().getTextResult(); noResponse = Text_Result.indexOf("html"); } while (noResponse >=0)</pre>
		Documentation:

Figure 23 Collaboration Rules Editor: set HttpClient TextData to be the input Rule

Collaboration Rules Editor - crGetHTTPResponse (Sandb	юх - Modified)			_ 8 ×
File Edit View Tools Help				
# h				
Source Events			Destination Events	
E ■ input [GenericOutEvent]				response [Envelope] ** 🕀 http [HttpClient] ** 🕀
O block ⇔ method ⊘ var Ita for 人	if () rule 🗮	switch → case I+	while Ita do 🖉 retur	n throw
() try ! catch () copy () datamap () li	list lookup () timestamp ()	uniqueid		
Business Rules				
crGetHTTPResponse crGetHTTPResponse crGetHTTPResponse crGetHTTPResponse crGetHTTPResponse creBoolean () set HttpClient TextData to be the input (ControlBlock) c) <u>Check errors and unmarshal the response</u> certain userInitialize userTerminate	t) and get UCCnet response	Rule Properties Description: Check errors. Rule: Int m = Text_Result.indexOf if (m <= 0) { int p = Text_Result.ind String result = Text_Re getresponse().unmarsh getresponse().send(); }	and unmarshal the response ("Error"); axOf("envelope communicationWersion"); sult.substring(p=1); al(STCTypeConverter.toByteArray(resull	(1));
		Documentation:		

Figure 24 Collaboration Rules Editor: Check errors and unmarshal the response Rule

- 20 For the **crGetRequestMessage** and **crPutResponseMessage** Collaboration Rules, all you need to do is select the **Pass Through** Service in the **Collaboration Properties** dialog box (**General** tab). No further configuration is required.
- 21 You need to create the **crGetRequestMessage** and **crPutResponseMessage** Collaboration Rules. To do so:
 - Create and name each Collaboration Rules component (**crGetRequestMessage** and **crPutResponseMessage**).
 - Open the **Collaboration Properties** dialog box (**General** tab) and select the **Pass Through** Service for both Collaboration Rules.
 - On the **Collaboration Properties** dialog box click **OK** to save both Collaboration Rules.

No further configuration is required.

4.4.8 Creating Collaborations

Collaborations are the components that receive and process Event Types, then forward the output to other e*Gate components or an external system.

Collaborations consist of the subscriber, which receives Events of a known type (sometimes from a given source), and the publisher, which distributes transformed Events to a specified recipient.

To create the Collaborations

- 1 In the e*Gate Schema Manager, select the Navigator's **Components** tab.
- 2 Open the host on which you want to create the Collaboration.
- 3 Select the Control Broker for this schema.
- 4 Select the **ewGetRequestMessage** e*Way to assign the Collaboration.
- 5 On the palette, click the **Collaboration** icon.
- 6 Enter the name (colGetRequestMessage) of the new Collaboration, then click OK.
- 7 Select the new Collaboration, then right-click to edit its properties.

8 The **Collaboration Properties** dialog box appears (see Figure 25).

🌐 Collaboration - colGetRequestMessage Properties _ 🗆 🗡 General colGetRequestMessage Collaboration Rules: crGetRequestMessage Ŧ New Edit Subscriptions: Event Type Source Add 🔜 <external> 📲 UCCnet (Required) Publications: Event Type Destination Priority 📲 UCCnet 😫 iqUCCnet 5 0K Cancel Apply Help

Figure 25 colGetRequestMessage Properties Dialog Box

- 9 From the **Collaboration Rules** list, select the Pass Through **Collaboration Rule** that you created previously (**crGetRequestMessage**) for this Collaboration.
- 10 In the **Subscriptions** area, click **Add** to define the input Event Type to which this Collaboration subscribes to (**UCCnet**). Check **Triggering**.
- 11 In the **Publications** area, click **Add** to define the output Event Type that this Collaboration publishes to (**UCCnet**). Select **Default**.
- 12 Click **OK** to close the dialog box.
- 13 Select the **ewPutResponseMessage** e*Way to assign the Collaboration.
- 14 On the palette, click the **Collaboration** icon.
- 15 Enter the name (**colPutResponseMessage**) of the new Collaboration, then click **OK**.
- **16** Select the new Collaboration, then right-click to edit its properties.

17 The **Collaboration Properties** dialog box appears (see Figure 26).

🜐 Collaboration - colPutResponseMessage Properties _ 🗆 🗡 General colPutResponseMessage Collaboration Rules: crPutResponseMessage Ŧ New Edit Subscriptions: Event Type Source Add 📲 UCCnet (Required) 🔏 colGetHTTPResponse Publications: Event Type Destination Priority 📲 UCCnet 🔜 <external> 5 OK. Cancel Apply Help

Figure 26 colPutResponseMessage Properties Dialog Box

- 18 From the **Collaboration Rules** list, select the Pass Through **Collaboration Rule** that you created previously (**crPutResponseMessage**) for this Collaboration.
- 19 In the **Subscriptions** area, click **Add** to define the input Event Type to which this Collaboration subscribes to (**UCCnet**). Check **Triggering**.
- 20 In the **Publications** area, click **Add** to define the output Event Type that this Collaboration publishes to (**UCCnet**). Select **Default**.
- 21 Click **OK** to close the dialog box.
- 22 Select the **mmGenerateHTTPRequest** e*Way to assign the Collaboration.
- 23 On the palette, click the **Collaboration** icon.
- 24 Enter the name (**colGenerateHTTPRequest**) of the new Collaboration, then click **OK**.

- 25 Select the new Collaboration, then right-click to edit its properties.
- **26** The **Collaboration Properties** dialog box appears (see Figure 27).

igure 27 Generater i i rikequest rioperties Dialog box	Figure 27	GenerateHT	TPRequest Pro	perties Dia	log Box
--	-----------	------------	----------------------	-------------	---------

∋eneral				
colGenerat	eHTTPRequest			
Collaboration Rules:				
crGenerateHTTPReque	st		New	Edit
Subscriptions:				
Instance Name	Event Type	Source		Add
request	🖬 UCCnet	💥 colGetReque	stMessage	
Publications:				
Publications:	Event Type	Destination	Priority	Add
Publications: Instance Name http	Event Type	Destination	Priority 5	Add
Publications: Instance Name http output	Event Type HttpClient GenericOutEvent	Destination ControlUCCnet	Priority 5	Add Delete
Publications: Instance Name http output	Event Type HttpClient GenericOutEvent	Destination IIII ecToUCCnet IIII iqControlBlock	Priority 5 5	Add Delete Advanced
Publications: Instance Name http output	Event Type HttpClient GenericOutEvent	Destination ControlUCCnet	Priority 5 5	Add Delete Advanced
Publications: Instance Name http output	Event Type HttpClient GenericOutEvent	Destination IIII ecToUCCnet IIII iqControlBlock	Priority 5 5	Add Delete Advanced
Publications: Instance Name http output	Event Type	Destination IIII ecToUCCnet iqControlBlock	Priority 5 5	Add Delete Advanced

- 27 From the **Collaboration Rules** list, select the Java **Collaboration Rule** that you created previously (**crGenerateHTTPRequest**) for this Collaboration.
- 28 In the **Subscriptions** area, click **Add** to enter the **request** instance name, select the **UCCnet** Event Type, and select the Collaboration **colGetRequestMessage** as the source.
- 29 In the **Publications** area, click **Add** to enter the **http** and **output** instance names, select the **HTTPClient** and **GenericOutEvent** Event Types, and select **ecToUCCnet** and **iqControlBlock** as the destinations.
- 30 Click **OK** to close the dialog box.
- 31 Select the **mmGetHTTPResponse** e*Way to assign the Collaboration.
- 32 On the palette, click the **Collaboration** icon.

- 33 Enter the name (colGetHTTPResponse) of the new Collaboration, then click OK.
- 34 Select the new Collaboration, then right-click to edit its properties.
- 35 The Collaboration Properties dialog box appears (see Figure 28).

Figure 28 colGetHTTPResponse Properties Dialog Box

		percies		
∋eneral				
	Response			
Collaboration Rules:				
crGetHTTPResponse			 New 	Edit
Subscriptions:				
Instance Name	Event Type	Source		Add
input	GenericOutEvent	t 🛛 🌌 colGenerate	eHTTPRequ	Delete
ļ				
Publications:				
Publications:	Event Type	Destination	Priority	Add
Publications: Instance Name response	Event Type	Destination	Priority 5	Add
Publications: Instance Name response http	Event Type	Destination iqResponse cFromUCCnet	Priority 5 5	Add
Publications: Instance Name response n http n	Event Type	Destination iqResponse iqResponse	Priority 5 5	Add Delete Advanced
Publications: Instance Name response n http	Event Type UCCnet	Destination	Priority 5 5	Add Delete Advanced
Publications: Instance Name response http	Event Type	Destination iqResponse cFromUCCnet	Priority 5 5	Add Delete Advanced

- 36 From the **Collaboration Rules** list, select the Java **Collaboration Rule** that you created previously (**crGetHTTPResponse**) for this Collaboration.
- 37 In the **Subscriptions** area, click **Add** to enter the **input** instance name, select the **GenericOutEvent** Event Type, and select the Collaboration **colGenerateHTTPRequest** as the source.
- 38 In the Publications area, click Add to enter the response and http instance names, select the UCCnet and HttpClient Event Types, and select iqResponse and ecFromUCCnet as the destinations.
- 39 Click **OK** to close the dialog box.

4.4.9 Running the Schema

To run the schema

• From the command line prompt, enter on a single line:

stccb -rh hostname -rs schemaname -un username
-up user_password -ln localhost_cb

Substitute the appropriate names for *hostname*, *username*, *schemaname*, and *user_password* as appropriate.

The schema components start automatically. When there are no more run-time messages, check the output file. If the schema is operating correctly, the final output directory contains the payload data.

Index

A

after installation UNIX 10 Windows 10

C

certification 7 Collaborations 46 commands (UCCnet command types) 15 components of library 6

E

ETD library 16

Η

HTTP(S) interface introduction 7 overview 7

implementation overview **18** installation UNIX **10** Windows **9** installed files and directories **16** intended reader **5**

Μ

messaging message components 12 message elements 13 overview 11 UCCnet messages 13

Ρ

pre-installation UNIX **10** Windows **9**

R

running a schema 51

S

sample schema components 21 operation 19 overview 19 setup 20 sample schema, creating 22 sample schema, importing 18 supported operating systems 8 system requirements external 8 regular 8

U

UCCnet overview introduction 5 using 6