SeeBeyond ICAN Suite

HTTP(S) eWay Intelligent Adapter User's Guide

Release 5.0.6

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.

© 2004 by 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 This work is protected as an unpublished work under the copyright laws.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 20040930084249.

Contents

Chapter 1

Introducing the HTTP(S) eWay

What's in This Chapter

1.1 About HTTP and HTTPS

HTTP

HTTP (hypertext transfer protocol) is the set of rules used for transferring files (text, graphic images, sound, and video) over the Web. When a user opens a Web browser, the user is indirectly making use of HTTP. HTTP is an application protocol that runs on top of the TCP/IP suite of protocols.

In addition to the files that it serves, every Web server contains an HTTP daemon—a program that waits for HTTP requests and handles them when they arrive. A Web browser is an HTTP *client*, sending requests to *server* machines. When the user enters a URL or clicks on a hypertext link, the browser builds an HTTP request and sends it to the IP address indicated by the URL. The HTTP daemon in the destination server machine receives the request and sends back the requested file or files associated with the request.

HTTPS

HTTPS (hypertext transfer protocol over secure socket layer—or HTTP over SSL) is a Web protocol that encrypts and decrypts user page requests as well as the pages that are returned by the Web server. HTTP uses port 443 instead of HTTP port 80 in its interactions with the lower layer TCP/IP. SSL uses a 40-bit encryption key algorithm, which is considered an adequate level of encryption for commercial exchange.

When an HTTPS request is sent by a browser—usually by clicking a link that begins with **https://**—the client browser encrypts the request and sends it to the Web server. The acknowledgement sent by the Web server is also sent using encryption, and is decrypted by the client browser.

Note: *This document uses the term HTTP(S) to refer to HTTP and HTTPS. In the case where a topic or concept does not apply to both, either HTTP or HTTPS is refered to specifically.*

1.2 About the HTTP(S) eWay

The HTTP(S) eWay enables eGate Integrator to communicate with client and server applications over the Internet using HTTP, either with or without the Secure Socket Layer (SSL).

1.2.1 HTTP Messages

An HTTP message has two parts: a request and a response. The message header is composed of a header line, header fields, a blank line, and an optional body (or data payload). The response is made up of a header line, header fields, a blank line, and an optional body (or data payload). HTTP is a synchronous protocol, that is, a client makes a request to a server and the server returns the response on the same socket.

1.2.2 Web Browser Cookies

A cookie is an HTTP header, which is a key-value pair in the header fields section of an HTTP message.

The **Set-Cookie** and **Cookie** headers are used with cookies. The **Cookie-request** header is sent from the server in request for cookies on the client side. An example of a **Cookie-request** header is:

```
Set-Cookie: sessauth=44c46a10; expires=Wednesday, 27-Sep-2000
03:59:59 GMT
```

In this example, the server requests that the client store the following cookie:

```
sessauth=44c46a10
```

Everything after the first semi-colon contains additional information about the cookie, such as the expiration date. When the eWay sees this header, it extracts the cookie *sessauth=44c46a10* and returns it to the server on subsequent requests. The eWay prepends a cookie header to the HTTP request, for example:

```
Cookie: sessauth=44c46a10
```

Each time the eWay sends a request to the same server during a session, the cookie is sent along with the request.

Cookie Expiration Date Checking

The HTTP(S) eWay checks time-limited cookies with expiration dates to ensure that they have not expired. If they have expired, the cookie is removed and is not resent to the originating server. As a result, the session state is removed.

The following standard expiration date formats are recognized by the HTTP(S) eWay:

```
"Sun, 06 Nov 1994 08:49:37 GMT"   ;RFC 822, updated by RFC 1123
"Sunday, 06-Nov-94 08:49:37 GMT";RFC 850, obsoleted by RFC 1036
"Sunday, 06-Nov-1994 08:49:37 GMT";RFC 1036
"Sun Nov  6 08:49:37 1994"    ;ANSI C's asctime()
```

If the expiration date is in another format, the eWay does not recognize the expiration date. Instead, it treats the cookie as if it does not have an expiration date.

1.2.3 GET and POST Methods

The **GET** method can be used to retrieve a page specified by the URL or to retrieve information from a form-based Web page by submitting URL-encoded key and name value pairs. In the latter case, the page must support the **GET** method.

The following example shows a URL-encoded query string:

http://.../bin/query?p=seebeyond+integrator

The URL specifies the search page and the name-value pair for the search. The question mark (?) indicates the beginning of the name-value pair encoding. In the previous example, the name portion of the query is "p," and the value to search is "seebeyond integrator." A query can consist of one or more of these name-value pairs.

Note: *See the official HTTP Specification for complete information.*

The **POST** method is more versatile, in that it supports form-based requests, as well as sending large amounts of data. The **POST** method does not have the size-limitation maximum of 255 or 1024 characters (depending on the Web server), which the **GET** method has. As with **GET**, the Web page must support the **POST** method in order to use **POST**.

Taking the previous URL as an example, if you specify the following URL:

http://. . ./bin/query

Then, you can specify the name-value pair separately. The HTTP client allows for the specification of the URL and n-number of value pairs via its methods.

1.2.4 Sample HTTP Exchange

To retrieve the file at the following URL:

http://www.myhost.com/path/file.html

First open a socket to the host **www.myhost.com**, port 80 (use the default port of 80 because none is specified in the URL). You can then send a request through a socket that looks like the following example:

```
GET /path/file.html HTTP/1.0  (Request Header Line)
User-Agent: HTTP(S)eWay       (Request Header field)
```

The server sends a response back through the same socket. The response could look like the following example:

```
HTTP/1.0 200 OK                (Response Header Line)
Date: Fri, 31 Dec 1999 23:59:59 GMT(Response Header Field)
Content-Type: text/html   (Response Header Field)
Content-Length: 1354           (Response Header Field)
[blank line here]
<html>                         (Response payload)
<body>
<h1>Happy New Millennium!</h1>
(more file contents)
  .
  .
  .
</body>
</html>
```

After sending the response, the server closes the socket.

1.3 What's New in This Version

Version 5.0.6 is a maintenance release of the HTTP(S) eWay. This release also adds support for SuSE Linux. For more information see **"Supported Operating Systems" on page 13**.

1.4 What's in This Document

This guide explains how to install, configure, and operate the SeeBeyond® Integrated Composite Application Network Suite™ (ICAN) HTTP(S) eWay Intelligent Adapter, referred to as the HTTP(S) eWay throughout this guide.

1.4.1 Organization of Information in This Book

This document includes the following chapters:

- **Chapter 1 "Introducing the HTTP(S) eWay"**: Provides an overview description of the product as well as high-level information about this document.
- **Chapter 2 "Installing the HTTP(S) eWay"**: Describes the system requirements and provides instructions for installing the HTTP(S) eWay.
- **Chapter 3 "Configuring the HTTP(S) eWay"**: Provides instructions for configuring the eWay.
- **Chapter 4 "Reviewing eInsight Projects"**: Describes how to use the HTTP(S) eWay with eInsight Business Process Manager and reviews an eGate Project that uses eInsight.

- **Chapter 5 "Reviewing Project With Collaborations"**: Describes how to implement the HTTP(S) eWay using a review of the sample Project, which uses Java-based Collaborations.
- **Chapter 6 "Understanding the HTTP(S) eWay OTD"**: Provides a description of the Object Type Definitions to be used with the HTTP(S) eWay.
- **Chapter 7 "Operating SSL"**: Explains the operation of the Secure Sockets Layer (SSL) feature available with the HTTP(S) eWay.
- **Chapter 8 "Using the OpenSSL Utility"**: Provides detailed information on how to use the OpenSSL utility.

1.4.2 Scope

This document describes the process of installing, configuring, and running the HTTP(S) eWay.

This document does not cover the Java methods exposed by this eWay. For information on the Java methods, download and view the HTTP(S) eWay Javadoc files from the Enterprise Manager.

1.4.3 Intended Audience

This guide is intended for experienced computer users who have the responsibility of helping to set up and maintain a fully functioning ICAN Suite system. This person must also understand any operating systems on which the ICAN Suite will be installed and must be thoroughly familiar with Windows-style GUI operations.

1.4.4 Writing Conventions

The following writing conventions are observed throughout this document.

Table 1 Writing Conventions

Text	Convention	Example
Button, file, icon, parameter, variable, method, menu, and object names.	**Bold** text	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**. ▪ Select the **logicalhost.exe** file. ▪ Enter the **timeout** value. ▪ Use the **getClassName()** method. ▪ Configure the **Inbound** File eWay.
Command line arguments and code samples	`Fixed` font. Variables are shown in ***bold italic***.	`bootstrap -p` ***`password`***
Hypertext links	**Blue** text	**http://www.seebeyond.com**

1.5 SeeBeyond Web Site

The SeeBeyond Web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.seebeyond.com

SeeBeyond Documentation Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

docfeedback@seebeyond.com

1.6 Related Documents

The following SeeBeyond documents provide additional information about the ICAN product suite:

- *eGate Integrator User's Guide*
- *SeeBeyond ICAN Suite Installation Guide*

Chapter 2

Installing the HTTP(S) eWay

What's in This Chapter

2.1 Supported Operating Systems

The HTTP(S) eWay is available on the following operating systems:

- Windows 2000, Windows XP, Windows Server 2003
- HP NonStop Server G06.22
- HP Tru64 V5.1A
- HP-UX 11.0, 11i (PA-RISC), and 11i v2.0 (11.23)
- IBM AIX 5.1L and 5.2
- Red Hat Enterprise Linux AS 2.1 (Intel x86)R
- Red Hat Linux 8 (Intel x86)
- Sun Solaris 8 and 9
- SuSE Linux Enterprise Server 8 (Intel x86)

2.2 System Requirements

The system requirements for the HTTP(S) eWay are the same as for eGate Integrator. Refer to the *SeeBeyond ICAN Suite Installation Guide* for a complete listing of system requirements. It is also helpful to review the **Readme.txt** file for additional requirements prior to installation.

Note: *To enable Web services, you must also install and configure eInsight.*

2.3 External System Requirements

The HTTP(S) eWay supports the following:

- HTTP versions 1.0 and 1.1
- RFCs 1945 (version 1.0), 2616 (version 1.1), and 2817 (TLS over version 1.1)
- The **get** and **post** Java methods
- Automatic URL redirection (automatic redirection occurs when the eWay receives a 301 status code)
- Single-sesstion request/reply scenarios in the default properties
- The Secure Sockects Layer (SSL) security feature (**Chapter 7 Operating SSL** on page 87 for details)
- Cookes in both HTTP and HTTPS modes

2.4 Installing the eWay Product Files

The installation process includes:

- Installing the ICAN Repository.
- Uploading products to the Repository (including the HTTP(S) eWay, documentation, sample files, and Javadocs).
- Downloading components (including the Enterprise Designer and Logical Host) from the Repository.
- Updating products in the Enterprise Designer using the Update Center Wizard.

To install the HTTP(S) eWay

1. Follow the instructions for installing ICAN in the *SeeBeyond ICAN Suite Installation Guide*.
2. After uploading **eGate.sar** to the Repository, upload the following additional product files:
   - **HTTPeWay.sar** (to install the HTTP(S) eWay)
   - **FileeWay.sar** (to install the File eWay, used in the sample Projects)
   - **HTTPeWayDocs.sar** (to install the HTTP(S) eWay documentation)

Note: *These files may not be located on the same installation disc as the **eGate.sar** file.*

To install the HTTP(S) eWay Samples and Javadocs

1. From the Documentation tab of the Enterprise Manager, click **HTTP(S) eWay** to view the list of files available for this product.
2. Click **Download Sample** to open the **HTTPeWaySample.zip** file.

3 Use WinZip to extract the sample files to the desired location.

4 Click Download Javadocs to open the **HTTPeWayJavadoc.zip** file.

5 Use WinZip to extract the Javadocs files to the desired location.

After you complete the process of installing the Repository, Logical Host, and Enterprise Designer (as described in the *SeeBeyond ICAN Suite Installation Guide*), refer to **Chapter 4** and **Chapter 5** for instructions on importing the sample project into your repository via the Enterprise Designer.

Chapter 3

Configuring the HTTP(S) eWay

What's in This Chapter

3.1 HTTP(S) eWay Properties Dialog Box

When you install the HTTP(S) eWay, a default properties template for the eWay is also installed. These default settings apply to all HTTP(S) eWays you use within your current Project or Business Process.

You can set properties for each individual eWay using the Enterprise Designer's eWay **Properties** dialog box. This user interface is available for the eWay in the following locations:

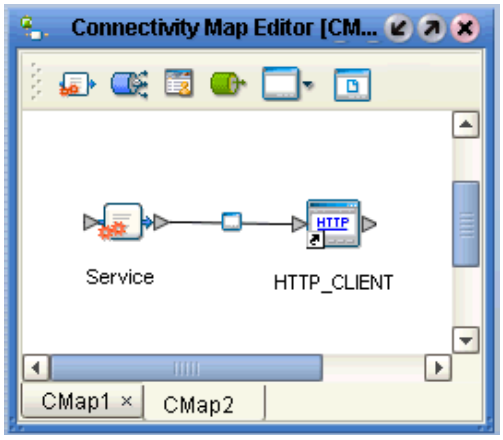
- Connectivity Map
- Project Environment

This section describes general procedures on how to change the default properties for the eWay. For details on these steps, see the *eGate Integrator User's Guide*.

To set properties for the HTTP(S) eWay on the Connectivity Map

1. From the eGate Enterprise Designer's **Project Explorer** create at least one Connectivity Map.
2. Create the desired external systems for your one or more Connectivity Maps.
3. Select the external application whose default eWay properties you want to change by clicking the **eWay** icon. This icon is located on the link between an **External Application** icon and a **Service** icon on the Connectivity Map canvas. See **Figure 1 on page 17**.

Figure 1 eWay Icon



The eWay **Properties** dialog box appears. **Figure 2 on page 18** shows the eWay's default properties available from the **Project Explorer** and Connectivity Map. You can use this window to modify the current eWay's properties settings.

4 Click **OK** then **Save All** to save your changes.

To set properties for the HTTP(S) eWay on the Project Environment

1 From the Enterprise Designer, create your external systems.

2 Click the **Environment Explorer** tab (at the bottom of the left pane).

3 Create an environment for your project, then create external systems on the Environment canvas to correspond to the systems you created using the **Project Explorer**.

4 Select the external system whose default eWay properties you want to change by right-clicking the desired system's icon (client or server) in the **Environment Explorer**.

The eWay **Properties** dialog box appears. **Figure 3 on page 23** shows the eWay's default properties available from the **Environment Explorer**. You can use this dialog box to modify the eWay properties associated with the current external system, including SSL-related properties.

5 Click **OK** then **Save All** to save your changes.

eWay Properties dialog box

- You can use properties set on the Environment to configure the eWay in either client mode, server mode, or both.
- Clicking the **Configuration** (Connectivity Map) or **Environment Configuration** folder in the left pane displays the properties group subfolder in the right pane. Click any subfolder to display the eWay's editable properties.
- Many of the entries allow you to enter text. Click the desired text box, then click the ellipsis (**...**) that appears, to open a dialog box for this purpose.

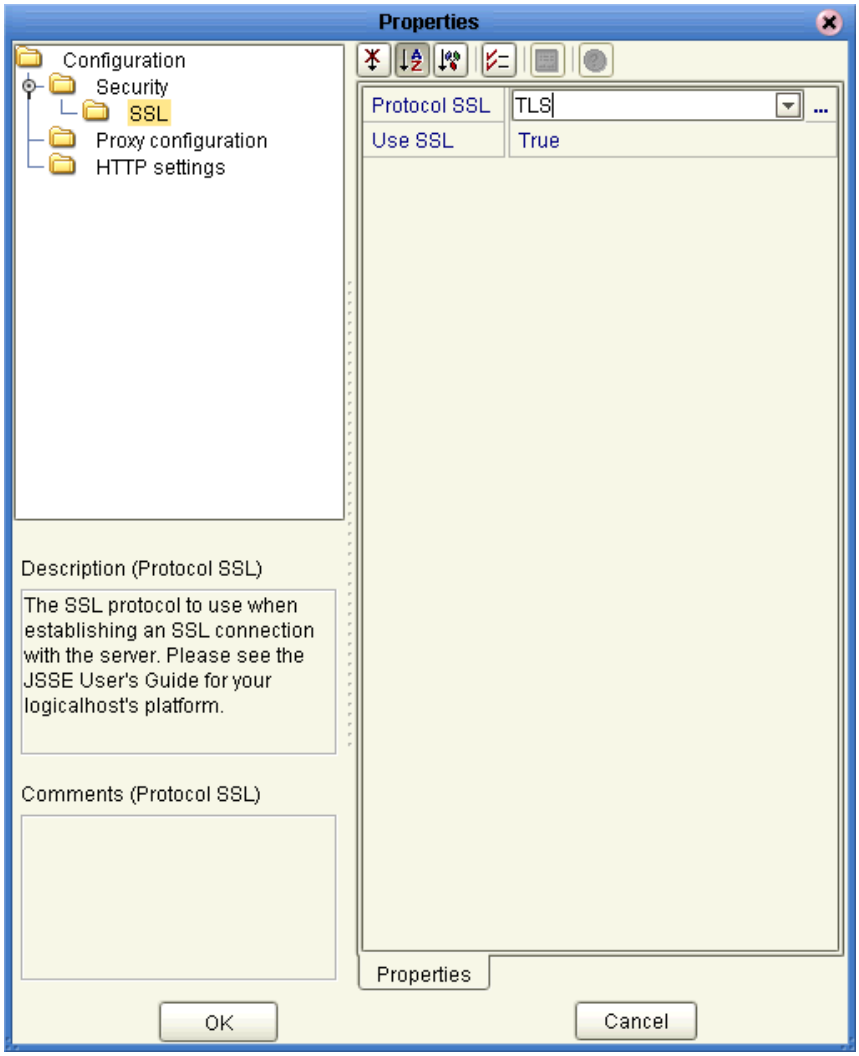
Note: *Even if you do not change the eWay's properties, you must open the **Properties** dialog box for the eWay and click **OK** to activate the eWay.*

The rest of this chapter explains all of the eWay's properties in detail, under the following sections:

3.2 Setting Properties on the Connectivity Map

This section explains in detail the eWay's editable properties (client and server modes) accessible via the eGate Enterprise Designer's **Project Explorer** and Connectivity Map. You can set these properties using the eWay **Properties** dialog box. See Figure 2.

Figure 2 eWay Properties Dialog Box: Settings on the Connectivity Map



These eWay **Project Explorer** properties are organized as follows:

3.2.1 Security/SSL

The **Project Explorer** editable properties in this section control the information required to set up the SSL connection via HTTP.

Protocol SSL

Description

The SSL protocol to use when establishing an SSL connection with the server. If the protocol is not set by this method, the default protocol type, **TLS** (Sun JSSE), is used.

Required Values

If you are using the default Sun JSSE provider, choose one of the following settings:

- **TLSv1**
- **TLS**
- **SSLv2**
- **SSLv3**
- **SSL**

If you are running the SeeBeyond Integration Server on AIX, choose one of the following settings:

- **SSL-TLS**
- **TLSv1**
- **TLS**
- **SSLv3**
- **SSLv2**
- **SSL**

For details on these settings, see the appropriate JSSE documentation.

Use SSL

Description

Specifies whether HTTP(S) connections are to be used. In other words, this property specifies whether SSL needs to be configured to use the HTTP(S) protocol. The default is **False**.

Required Values

True or **False**.

3.2.2 Proxy Configuration

The **Project Explorer** editable properties in this section specify the information required for the eWay to access the external systems through a proxy server.

Use the **Proxy Configuration** settings in the client HTTP(S) eWay properties, when setting the desired URL dynamically within a Collaboration (Java) or Business Process.

Note: *It is a known behavior of the Java Virtual Machine (JVM) to bypass an invalid proxy server through a local connection. As a result, you may still get a response, even if the proxy setting is invalid. This false response only happens with an HTTP connection. An HTTP(S) connection ensures authenticated handshaking from the proxy.*

Proxy host

Description

The host name of the HTTP proxy. This specifies the HTTP(S) proxy host to which requests to an HTTP server or reception of data from an HTTP server may be delegated to a proxy. This sets the proxy port for secured HTTP connections.

Required Values

A valid HTTP(S) proxy host name.

Proxy password

Description

Specifies the password required for accessing the HTTP(S) proxy.

Required Values

The appropriate password.

Important: *Be sure to enter a value for the **Proxy username** properties before entering this property.*

Proxy port

Description

The port of the HTTP(S) proxy. This specifies the HTTP(S) proxy port to which requests to an HTTP server or reception of data from an HTTP server may be delegated to a proxy. This sets the proxy port for secured HTTP connections.

Required Values

A valid HTTP(S) proxy port. The default is **8080**.

Proxy username

Description

Specifies the user name necessary for authentication to access the proxy server.

Required Values

A valid user name.

Additional Information

The user name required by URLs that require HTTP basic authentication to access the site.

Important: *Be sure to enter a value for this property before you enter a value for the* ***Proxy password*** *properties.*

3.2.3 HTTP Settings

This section contains the **Project Explorer** editable properties used by HTTP.

Caution: *Calling the* ***clear()*** *method in the Collaboration Editor (Java) clears all properties in this* ***HTTP Settings*** *section. Once the properties have been cleared, you must manually rebuild the header and payload sections of the Request message in the Transformation Designer.*

Accept type

Description

The default **Accept type** header value to include when sending a request to the server.

Required Values

A string. For example **text/html, text/plain, text/xml,** and so on. The default is **text/***

Allow cookies

Description

Specifies whether cookies sent from servers are allowed to be stored and sent on subsequent requests. If cookies are not allowed, sessions are not supported.

Required Values

True or **False**. The default is **True**.

3.2.4 HTTP Server External Configuration

The **Project Explorer** editable properties in this section control information required specifically to set up the HTTP(S) server.

servlet-url

Description

Allows you to enter the last path component of the HTTP(S) server servlet URL. This URL is the one the client uses to access the server.

This property must be the servlet name, for example, **HttpServerServlet**. The total URL is made up of several components, including the Project deployment name and the value entered for this property.

An example of a complete servlet URL is:

http://localhost:18004/Deployment1_servlet/HttpServerServlet

Where:

- **localhost**: The name of the machine your current Logical Host is running on.
- **18004**: The port number (in this case, the SeeBeyond Integration Server port number).
- **Deployment1_servlet**: The name of your current Project's Deployment Profile concatenated with **_servlet**.
- **HttpServerServlet**: The servlet name, that is, the **servlet-url** property.

Note: *Set the port number based on the SeeBeyond Integration Server properties. By default, it is 18004, but it can be modified by the user. Set the SeeBeyond Integration Server properties using the* ***Environment Explorer****. See the* ***eGate Integrator User's Guide*** *for details.*

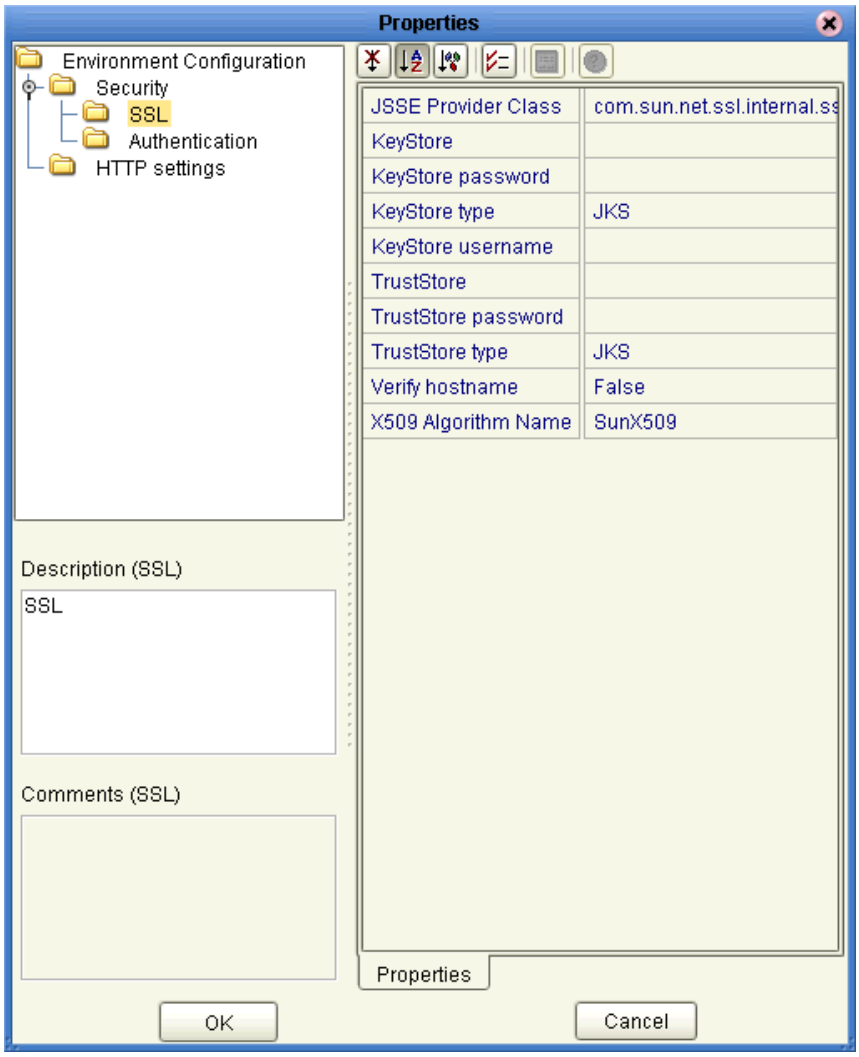
Required Values

A valid URL.

3.3 Setting Properties on the Project Environment

This section explains in detail the eWay's editable properties (client and server modes) accessible via the eGate Enterprise Designer's **Environment Explorer**. You can set these properties using the eWay **Properties** dialog box. See Figure 3.

Figure 3 eWay Properties Dialog Box: Settings on the Project Environment



These eWay **Environment Explorer** properties are organized as follows:

- **"Security/SSL" on page 24**
- **"Security/Authentication" on page 29**
- **"HTTP Settings" on page 30**

3.3.1 Security/SSL

The **Environment Explorer** editable properties in this section are used to set the basic security features for SSL, including CA certificates.

JSSE Provider Class

Description

Specifies the fully qualified name of the JSSE provider class. For more information, see the Sun Java Web site at:

http://java.sun.com/

Required Values

The name of a valid JSSE provider class; the default is:

com.sun.net.ssl.internal.ssl.Provider

If you are running the SeeBeyond Integration Server on AIX, specify:

com.ibm.jsse.IBMJSSEProvider

See the **procedure on page 27** for details on running the SeeBeyond Integration Server on AIX.

KeyStore

Description

Specifies the default KeyStore file. The keystore is used for key/certificate management when establishing SSL connections.

Required Values

A valid package location; there is no default.

KeyStore password

Description

Specifies the default KeyStore password. The password is used to access the KeyStore used for key/certificate management when establishing SSL connections; there is no default.

KeyStore type

Description

Specifies the default KeyStore type. The keystore type is used for key/certificate management when establishing an SSL connection. If the default KeyStore type is not set by this method, the default KeyStore type, JKS, is used.

KeyStore username

Description

The username for accessing the keystore used for key/certificate management when establishing SSL connections.

Note: *If the keystore type is PKCS12 or JKS, the keystore username properties is not used. PKCS12 and JKS keystore types require passwords for access but do not require user names. If you enter a value for this property, it is ignored for PKCS12 and JKS.*

TrustStore

Description

Specifies the default TrustStore. The TrustStore is used for CA certificate management when establishing SSL connections.

Required Values

A valid **TrustStore** name; there is no default.

TrustStore password

Description

Specifies the default TrustStore password. The password is for accessing the TrustStore used for CA certificate management when establishing SSL connections.

Required Values

A valid **TrustStore** password; there is no default.

TrustStore type

Description

The TrustStore type of the TrustStore used for CA certificate management when establishing SSL connections. If the TrustStore type is not set by this method, the default TrustStore type, **JKS**, is used.

Required Values

A valid **TrustStore** type.

Verify hostname

Description

Determines whether the host name verification is done on the server certificate during the SSL handshake.

You can use this property to enforce strict checking of the server host name in the request URL and the host name in the received server certificate.

Required Values

True or **False**; the default is **False**.

Additional information

Under some circumstances, you can get different Java exceptions, depending on whether you set this property to **True** or **False**. This section explains what causes these exceptions.

For example, suppose the host name in the URL is **localhost**, and the host name in the server certificate is **localhost.stc.com**. Then, the following conditions apply:

- If **Verify hostname** is set to **False**:

  Host name checking between the requested URL and the server certificate is turned *off*.

  You can use an incomplete domain host name, for example, **https://localhost:444**, or a complete domain host name, for example, **https://localhost.stc.com:444**, and get a positive response in each case.

  See the next section "Logical Host Java SDK versions" for details.

- If **Verify hostname** is set to **True**:

  Host name checking between the requested URL and the server certificate is turned *on*.

Note: *If you use an incomplete domain host name, for example,* ***https://localhost:444****, you can get the exception* ***java.io.IOException: HTTPS hostname wrong****.*

You must use a complete domain host name, for example, **https://localhost.stc.com:444**.

Note: *If the Java Software Developer's Kit (SDK) version used by the Logical Host and the corresponding Logical Host property setting do not match, you can get the exception* ***java.lang.ClassCastException****.*

Logical Host Java SDK versions

The HTTP(S) eWay supports both the Java SDK versions 1.3 and 1.4 with a Logical Host. Before activating a Project using this eWay, you must specify which version of the Java SDK the current Logical Host is using. For information on how to set the **Java SDK Version** property in the Logical Host, see the *eGate Integrator User's Guide*.

For example, for a Logical Host deployed with a Windows operating system, you must set the **Java SDK Version** property to **Version 1.4** before you activate the Project. If you choose **Version 1.3** (the default), and the Logical Host is running on version 1.4, you can get the following exception:

```
Problem with loading HostnameVerifier class instance for
com.stc.connector.httpadapter.http.NoHostNameVerifier:
java.lang.ClassCastException
```

If the Java SDK version selected for the Logical Host during activation is different from the JRE version of the Logical Host where the Project is deployed, you can also get the following exception:

```
Failed to initialize the EwayConnection instance of type
com.stc.connector.httpadapter.eway.HTTPEwayConnection; Exception:
Failed to create a physical connection to EIS
HTTP.java.lang.ClassNotFoundException: ACL: Class
com.stc.connector.httpadapter.http.jdk14.HttpClientAPIJDK14 not
found.
```

The Logical Host Java SDK property also applies to the JVM version. If you are running a JVM version 1.3 Logical Host, the **Java SDK Version** property *must* be set to **Version 1.3**. If you are running a JVM version 1.4 Logical Host, this property *must* be set to **Version 1.4**.

X509 Algorithm Name

Description

Specifies the X509 algorithm name to use for the trust and key manager factories.

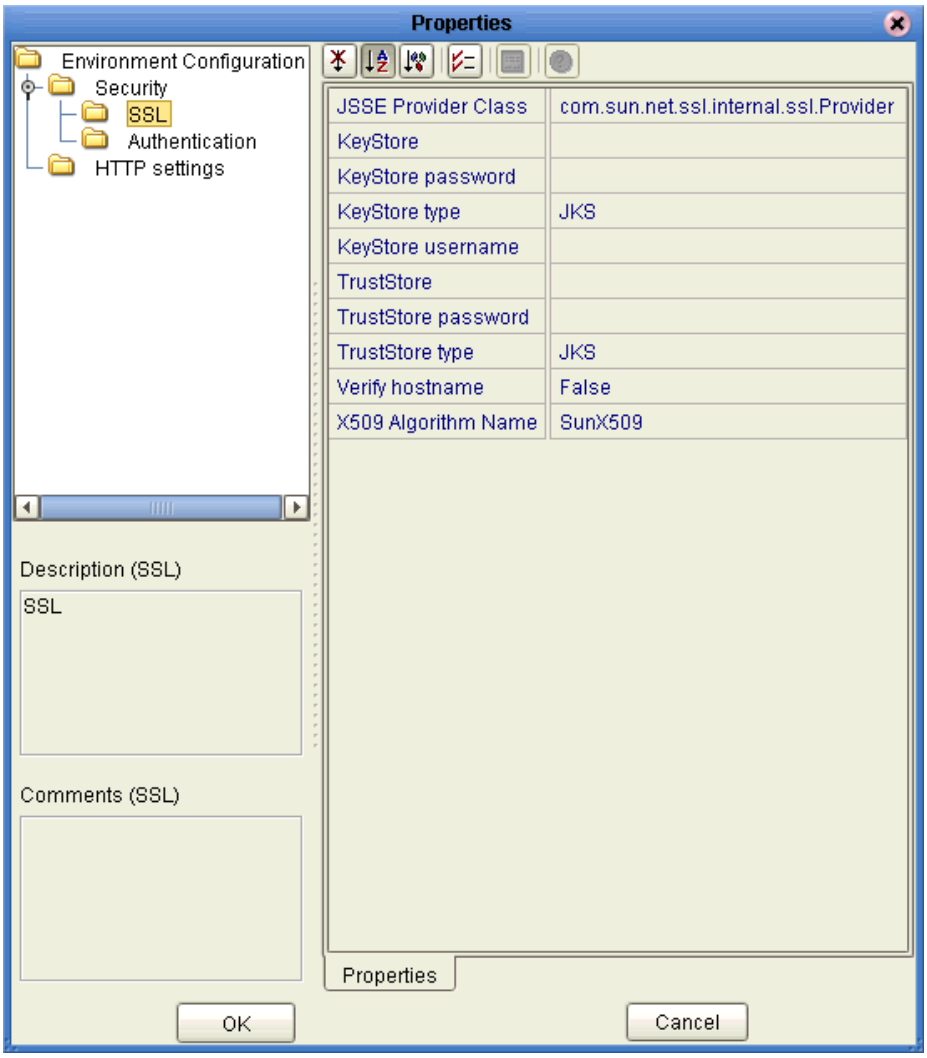
Required Values

The name of a valid X509 algorithm; the default is **SunX509**. If you are running the SeeBeyond Integration Server on AIX, specify **IbmX509**.

To set properties for an HTTP(S) external system for AIX

1. In the **Environment Explorer**, right-click the HTTP(S) external system and open its properties dialog box.
2. Under **Environment Configuration**, open **Security** and click **SSL**. See **Figure 4 on page 28**.

Figure 4 HTTP(S) Environment Security\SSL Default Properties

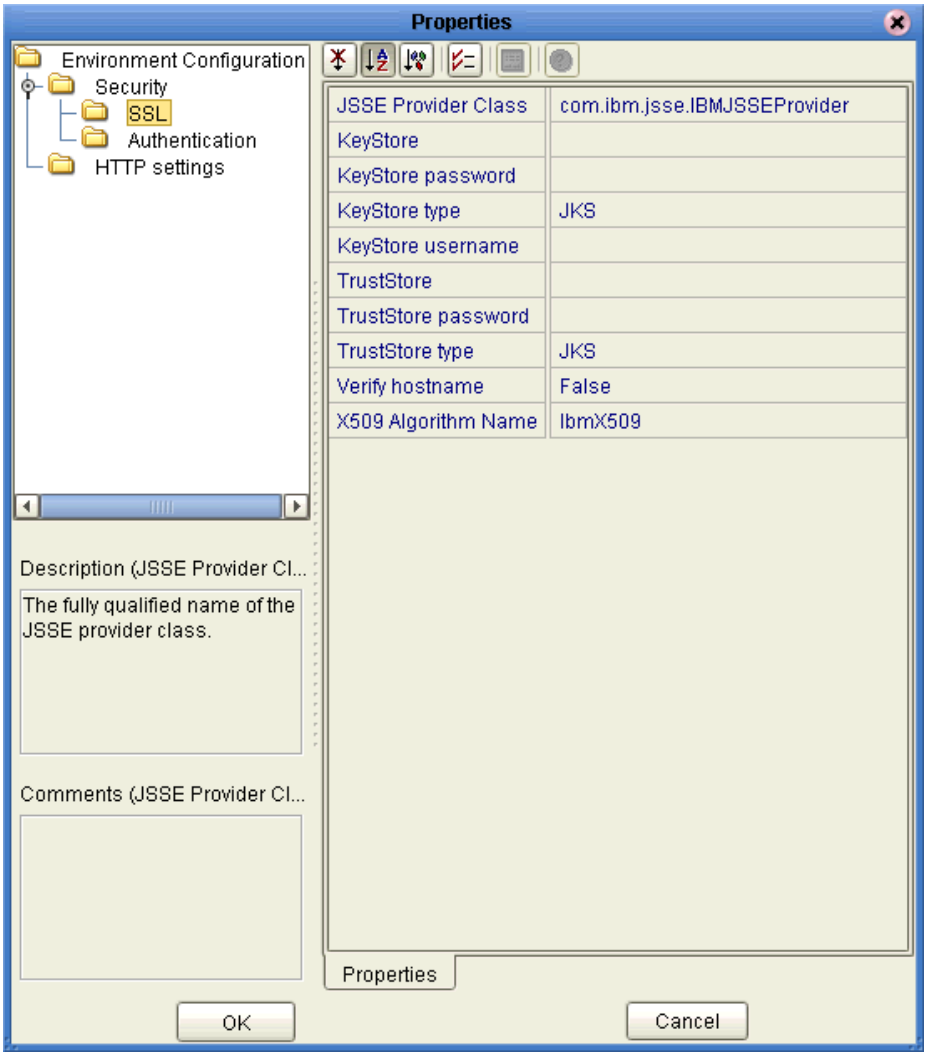


3 Make the following changes:

- For **JSSE Provider Class**, change the default to: **com.ibm.jsse.IBMJSSEProvider**
- For **X509 Algorithm Name**, change the default to: **IbmX509**

See **Figure 5 on page 29**.

Figure 5 HTTP(S) Environment Security\SSL AIX Properties



4 Click OK:

Note: *Enter these strings exactly as given in the previous procedure; for example, "IBMX509" is incorrect.*

3.3.2 Security/Authentication

The **Environment Explorer** editable properties in this section are used to perform HTTP authentication.

Http password

Description

Specifies the password used for authenticating the web site specified by the URL.

Required Values

A valid password.

Important: *Be sure to enter a value for the **Http username** properties before entering this property.*

Http username

Description

Specifies the user name for authenticating the web site specified by the URL.

Required Values

A valid user name.

Important: *Enter a value for this property before you enter a value for the **Http password** properties.*

3.3.3 HTTP Settings

This section contains the **Environment Explorer** editable properties used by HTTP.

Content type

Description

The default **Content type** header value to include when sending a request to the server.

Required Values

A string; there is no default.

Encoding

Description

The default encoding used when reading or writing textual data. The default is **ASCII**.

URL

Description

Specifies the default URL to be used for establishing an HTTP or HTTP(S) connection. When a URL is not assigned to the HTTP OTD, the default value is used as the URL for both the GET and POST commands. See **"GET and POST Methods" on page 9**.

If "https" protocol is specified, SSL must be enabled. See **"Protocol SSL" on page 19**.

Required Values

A valid URL. The default is **http://www.seebeyond.com**

Additional Information

You must include the full URL. For example,

http://www.seebeyond.com

or

http://google.yahoo.com/bin/query

If using GET functionality, you can provide the properties, using encoded query string notation. For example (all on one line):

http://www.ee.cornell.edu/cgi-bin/cgiwrap/~wes/pq?FirstName=John&LastName=Doe

Note: *For international URLs, be sure the targeting URL supports the encoding used in this property. A list of the character encoding supported by the Java 2 platform is at the Sun Web site:*

http://java.sun.com

3.4 Setting Web Connector Thread Properties

Before you run the eGate Logical Host, you must set the Web connector thread properties for the SeeBeyond Integration Server. These properties allow you to set up the correct performance of the HTTP(S) eWay in the server mode.

Web connector threads operate as follows:

- At server start-up time, the Web connector creates a number of request-processing threads. Each incoming request requires a thread for the duration of that request.
- If more simultaneous requests are received than can be handled by the currently available request-processing threads, additional threads are created. This number is limited by the configured maximum.
- If still more simultaneous requests are received, they are queued inside the server socket created by the Web connector, up to the configured maximum. Any further simultaneous requests receive "connection refused" errors, until resources are available to process the requests.

Note: *For more information, visit the Apache Tomcat server, version 4.1, Web site.*

In eGate, such errors appear in the log file for the server mode eWay component, as shown in the following example:

```
2003-12-12 16:32:14,015 INFO  [Thread-8]
[com.stc.sms.mbeans.process.ProcessProxy] [] IntegrationSvr1 ERR>
Dec 12, 2003 4:32:14 PM org.apache.tomcat.util.threads.ThreadPool log
Full
2003-12-12 16:32:14,015 INFO  [Thread-8]
[com.stc.sms.mbeans.process.ProcessProxy] [] IntegrationSvr1 ERR>
SEVERE: All threads are busy, waiting. Please increase maxThreads or
check the servlet status20 20
```

JVM implementations almost always use native threads, which means the thread limit is close to the process limit of what the operating system can handle. For example, in Windows, this limit is approximately 128 threads per process.

To prevent these errors, you must reset the following SeeBeyond Integration Server properties:

- **Accept Count**
- **Maximum Required Processing Threads**

Note: *It is recommended that you make these changes before you run any Logical Host on the current SeeBeyond Integration Server.*

To reset the necessary properties

1. From the eGate Enterprise Designer's **Environment Explorer**, click the desired SeeBeyond Integration Server and choose **Properties** from the pop-up menu.

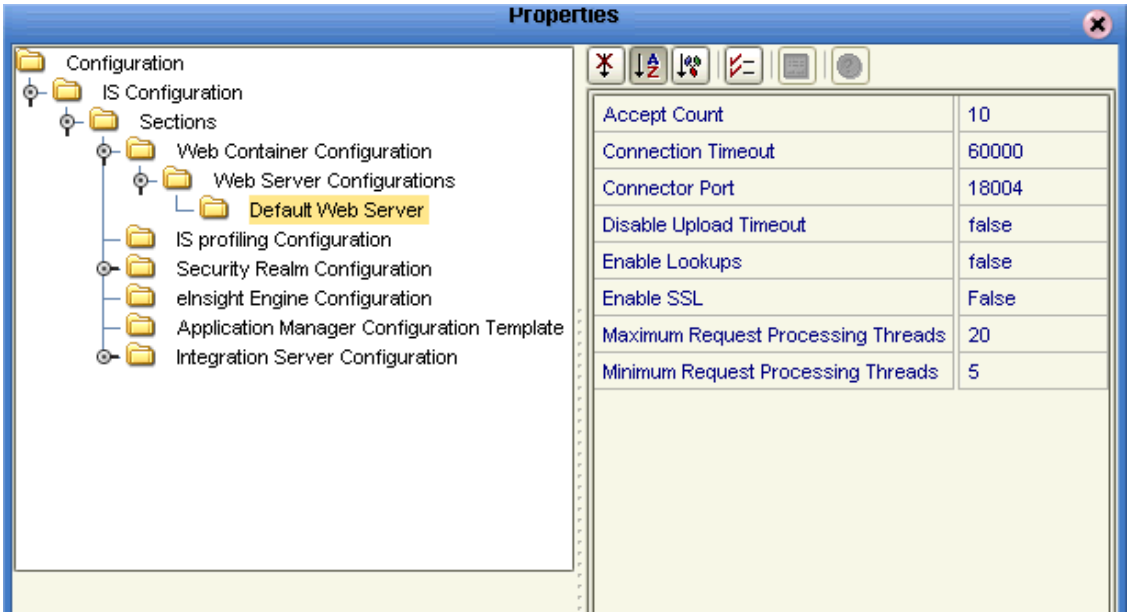
   The SeeBeyond Integration Server Properties dialog box appears.

2. Under the configuration folder in the left pane, expand the following folders:

   IS Configuration\Sections\Web Container Configuration\ Web Server Configurations\Default Web Server

   See Figure 6.

Figure 6 Logical Host Properties: Web Connector Threads



3 Reset the to **Accept Count** to 5000 and the **Maximum Required Processing Threads** to -1 as follows:

- **maxProcessors** to -1
- **acceptCount** to 5000

4 Close the Properties dialog box window.

For complete information on how to set the SeeBeyond Integration Server properties in eGate, see the *eGate Integrator User's Guide*.

Chapter 4

Reviewing eInsight Projects

This chapter describes how to use the HTTP(S) eWay with SeeBeyond ICAN Suite's eInsight Business Process Manager. It also reviews eGate Projects that operate using eInsight Business Processes.

Note: *You must have the* ***eInsight.sar*** *file installed to use the features explained in this chapter. See the* ***SeeBeyond ICAN Suite Installation Guide*** *for complete installation procedures.*

This chapter assumes that you are already familiar with eGate and eInsight concepts and that you understand the basics of creating a Project using the eGate Enterprise Designer.

For a complete explanation of eGate/eInsight terminology, operation, and concepts, see the *eGate Integrator User's Guide* and the *eInsight Business Process Manager User's Guide*.

What's in This Chapter

4.1 Project Canvas

Each eGate Project is created using the Enterprise Designer's Project canvas. The Project canvas contains windows that represent the various stages of your Project. The types of windows in your Project canvas area include:

- **Connectivity Map Canvas**: Contains the eGate business logic components, such as Collaborations, Topics, Queues, and eWays, that you include in the structure of the Project.
- **OTD Editor**: Contains the source files used to create Object Type Definitions (OTDs) to use with a project.
- **Business Process Canvas**: Allows you to use eInsight's Business Process features.

4.2 eInsight Engine and Components

You can set up and deploy an eGate Integrator component using eInsight. Once you have associated the desired component with a Business Process, the eInsight engine can automatically invoke that component during run time, using a Business Process Execution Language (BPEL) interface.

Examples of eGate components that can interface with eInsight in this way are:

- Java Messaging Service (JMS)
- Object Type Definitions (OTDs)
- An eWay
- eGate Services

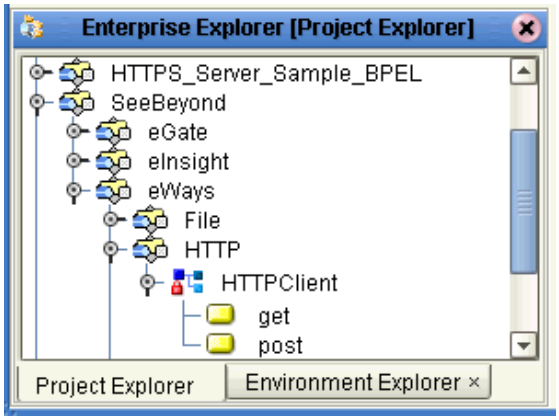
Using the eGate Enterprise Designer and its eInsight canvas, you can add a desired operation to a Business Process, then associate that process with an eGate component, for example, a Service. In the Enterprise Designer, associate the Business Process and Service icons using drag-and-drop operations.

See the *eInsight Business Process Manager User's Guide* for details.

4.3 HTTP(S) eWay With eInsight

You can add HTTP(S) eWay objects to an eInsight Business Process during the system design phase. To make this association, select the desired **get** or **post** operation under the eWay in the Enterprise Explorer and drag it onto the eInsight Business Process canvas. In turn, you can activate the Business Process in eGate by dragging it onto a Service or directly onto the canvas. See Figure 7.

Figure 7 HTTP(S) eWay get and post Operations



At run time, the eInsight Engine is able to invoke each of the steps in order as set up in the Business Process. Using the engine's BPEL interface, eInsight in turn invokes the HTTP(S) eWay operation, as well as any other eWays in the current Project.

Server Mode Operation

Instead of **get** and **post**, the eWay's server mode process **Request** operation. In addition these operations, the eInsight **Business Rule Designer** allows you to perform a variety of actions, represented by nodes in the **Output** and **Input** panes.

The actions allowed vary, depending on whether you are using the **Receive** or **Reply** functions. These actions allow you to perform operations in the same way as making calls using Java methods.

Table 2 explains the functions of these nodes.

Table 2 Receive: Business Rule Designer Output Nodes

Node Name	Description
authType	Gets or sets the name of the authentication scheme used to protect the servlet.
byteArray	Gets or sets the contents of the message as a byte array.
characterEncoding	Gets or sets the name of the character encoding used.
contentLength	Gets or sets the length, in bytes, of the message body.
contents	Sets the contents of the reply.
contentType	Gets or sets the MIME type of the body of the message, or null if the type is not known.
contextPath	Gets or sets the portion of the message URI that indicates the context of the message.
errorStatusCode	Gets or sets the error status code.
errorStatusMsg	Gets or sets the error status message.
isRequestedSessionIdFrom Cookie	Checks or sets whether the requested session ID came in as a cookie.
isRequestedSessionIdFrom URL	Checks or sets whether the requested session ID came in as part of the request URL.
isRequestedSessionIdValid	Checks or sets whether the requested session ID is still valid.
isSecure	Gets or sets a boolean indicating whether this message was made using a secure channel, such as HTTPS.
method	Gets or sets the name of the HTTP method with which this message was made; for example, GET, POST, or PUT.
name (WebHeaderList)	Gets or sets the name of the current Web header list.
name (WebParameterList)	Gets or sets the value of a request parameter as a String, or null if the parameter does not exist.
pathInfo	Gets or sets any extra path information associated with the URL the client sent when it made this message.
pathTranslated	Gets or sets any extra path information after the servlet name but before the query string.

Table 2 Receive: Business Rule Designer Output Nodes

Node Name	Description
protocol	Gets or sets the name and version of the protocol the message uses in the form protocol/majorVersion.minorVersion, for example, HTTP/1.1.
queryString	Gets or sets the query string that is contained in the message URL after the path.
redirectLocation	Gets or sets the URL to which the client is to be redirected.
remoteAddr	Gets or sets the Internet Protocol (IP) address of the client that sent the message.
remoteHost	Gets or sets the fully qualified name of the client that sent the message.
remoteUser	Gets or sets the log-in of the user making this request, if the user has not been authenticated.
requestedSessionId	Gets or sets the session ID specified by the client.
requestURI	Gets or sets the part of this message's URL from the protocol name up to the query string in the first line of the HTTP message.
requestURL	Gets or sets the reconstructed URL the client used to make the request.
scheme	Gets or sets the name of the scheme used to make this request; for example HTTP, HTTPS, or FTP.
serverName	Gets or sets the host name of the server that received the message.
serverPort	Gets or sets the port number on which this message was received.
servletPath	Gets or sets the part of this request's URL that calls the servlet.
status	Sets the status of the reply.
text	Gets or sets the contents of the message as a string.
values (WebHeaderList)	Gets or sets an array String objects containing all the values contained in the current Web header.
values (WebParameterList)	Gets or sets an array String objects containing all the values the given request parameter has, or null if the parameter does not exist.

Sample Projects

The HTTP(S) eWay has the following eInsight Business Process samples:

- **HTTPS_Client_Project_BPEL**: Client mode; see **"Reviewing the Business Process Project: Client" on page 38**.
- **HTTP_Server_Project_BPEL**: Server mode; see **"Building the Business Process Project: Server" on page 62**.

4.4 Reviewing the Business Process Project: Client

This section describes how to implement the HTTP(S) eWay using the eGate Project client sample with an eInsight Business Process, which is included on your installation CD-ROM. The sample is named **HTTPS_Client_Project_BPEL**. It allows you to observe an end-to-end data-exchange scenario involving eGate and the HTTP(S) eWay.

This section also explains how to implement this sample Project, including the HTTP(S) eWay in client mode. You can also use the review given in this chapter to create your own Projects based on the sample provided.

4.4.1 Client Sample Project: Overview

The client HTTP(S) eWay sample Project with an eInsight Business Process demonstrates how the HTTP(S) eWay uses the GET and POST commands to request and receive data from a specific Web site.

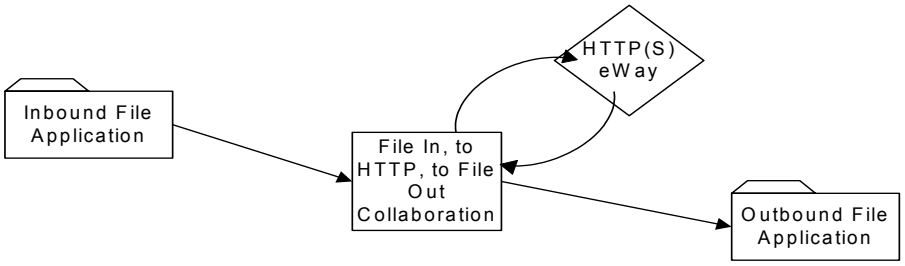
The data result is received from the Web site and is sent to a text file written to an external system via an outbound File eWay, to show the returned data and to confirm that the Project is operating correctly.

The Project has the following outputs:

- **GET Operations**: Returns the retrieved data in an .**html** file.
- **POST Operations**: Posts a name/value pair to a form and writes the same information to an .**html** file, to confirm the posting.

Figure 8 shows the flow of the sample HTTP(S) eWay Project.

Figure 8 HTTP(S) eWay Sample Project



The location of input and output files are defined by the File eWay properties. By default, the inbound File eWay reads from **c:\temp\input*.txt**. The default is changed for the Project's outbound File eWay, which sends the resulting data to **c:\temp\output%d.html** (**%d** represents the serial index starting with integer 0).

4.4.2 Importing Sample Projects

Before you can view or work with a sample Project, you must first import it into eGate, using the Enterprise Designer.

Note: *The sample* ***.zip*** *file you first download may contain more than one Project and/or additional files. If this is the case, you must unzip this file first, find the desired Project file, then import the Project file. For details on how to download this file, see the* ***SeeBeyond ICAN Suite Installation Guide****.*

The container file you are looking for is **HTTP_eWay_Sample.zip**. The Project file names are:

- **HTTP_Project_BPEL.zip**: eInsight Project
- **HTTP_Project_JCE.zip**: Collaboration (Java) Project

You can name the imported Projects as desired.

To import a sample Project

1. From the Enterprise Designer's **Project Explorer** pane, right-click the desired Repository and select **Import**.
2. On the **Import Manager** window, browse to the directory that contains the sample Project **.zip** file.
3. Select the sample Project file and click **Open**.
4. Click **Import**. If the import was successful, click **OK** on the **Import Status** dialog box.
5. Close the **Import Manager** window and select **Refresh All from Repository** from the shortcut menu.

Important: *An imported Project does not contain an environment or a deployment profile. After importing a Project, you must use the Enterprise Designer to create these functions for the Project. See* **"Creating the Project's Environment" on page 60** *and* **"Deploying a Project" on page 74***. For additional information, see the* ***eGate Integrator User's Guide*** *and* ***SeeBeyond ICAN Suite Deployment Guide****.*

You must check out the major eGate components before you can change them.

4.4.3 Client Sample Project Components and Operation

The HTTP(S) eWay client sample Project demonstrates how the HTTP(S) eWay processes information from an HTTP(S) system via an eInsight Business Process. Resulting or confirming information is then written to a text file. This scenario is shown in **Figure 8 on page 38**.

Project Components

The client Project has the following components:

- External file system (inbound): **FileIn**
- Inbound File eWay
- Business Process Service for processing data: **HttpBpelService** (created from the **Service** icon)
- HTTP(S) eWay
- HTTP client external application: **HTTP_CLIENT**
- Outbound File eWay
- External file system (outbound): **FileOut**

Project Operation

The client Project operates as follows:

- **FileIn**: The external file system that provides instructions to the inbound File eWay; this eWay gets a text file containing the instructions and passes them to a Business Process, **HttpBpelService**.
- **HttpBpelService**: Sends instructions to the desired HTTP system via the HTTP(S) eWay. **HttpBpelService** also receives the information from the HTTP(S) system, via the HTTP(S) eWay, then sends it to a File eWay, **FileOut**.
- **HTTP_CLIENT**: The HTTP client external application or system; the HTTP(S) eWay handles inbound and outbound communication with this system.
- **FileOut**: The external file system that receives the information via HTTP; another File eWay writes the received information to a text file on this system.

Input and Output Data

The HTTP(S) eWay Project uses the following input/output data files:

- **Get_Sample.xml**
- **Post_Sample.xml**
- **MultipleData_In.dtd**

These files have the following content:

GET Command: Get_Sample.xml

The input data file for the GET command is:

```
<website>
    <method>GET</method>
    <url>http://<rep host>:<rep port>/examples/servlet/
        HelloWorldExample</url>
    <data/>
</website>
```

POST Command: Post_Sample.xml

The input data file for the POST command is:

```
<website>
    <method>POST</method>
    <url>http://<rep host>:<rep port>/examples/servlet/
        RequestParamExample</url>
    <data><name>firstname</name><value>MyFirstName</value></data>
    <data><name>lastname</name><value>MyLastName</value></data>
</website>
```

Sample DTD: MultipleData_In.dtd

The eGate OTD wizard is used to create a DTD-based OTD. The input data file specifies an URL for HTTP commands. The XML DTD code for this sample input data file is:

```
<!ELEMENT website (method, url, data*)>
<!ELEMENT method (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT data (name?, value?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT value (#PCDATA)>
```

The **MultipleData_In.dtd** file defines the following elements:

- **Method**: Defines whether the file is for a GET or POST command.
- **URL**: Defines the address of the target HTTP server.
- **Data**: Stores the name/value pair used in the POST command; you can use as many name/value pairs as you need.

Instead of getting and posting relative to an external Internet site, this Business Process sample uses the eGate Integration Server and does these operations internally. If external Internet access is available, you can use that URL in the URL tag.

4.4.4 Creating a New Project

To create and name a new Project in the eGate Enterprise Designer

1. Start the Enterprise Designer.
2. Select the Enterprise Explorer's **Project Explorer** tab, to show the **Project Explorer** pane (left pane).
3. Select the **Repository** icon on the Project Explorer tree.
4. Right-click the Repository and select **New Project** on the pop-up menus**.**

   A new Project appears on the **Project Explorer** tree, named **Project1**.
5. Double-click on the Project name and rename the Project, for this sample, **HTTPS_Client_Project_BPEL**.

4.4.5 Using OTDs

An OTD contains a set of rules that define an object, which encodes data as it travels through eGate. OTDs are used as the basis for creating Business Processes for a Project. The HTTP(S) eWay provides HTTP OTD for this purpose. See **Chapter 6** for complete information on how to use this OTD.

User-defined OTD

You can use the OTD wizard to create an eGate User-defined OTD. See the *eGate Integrator User's Guide* for a complete explanation of how to create a User-defined OTD.

Creating OTDs Using the OTD Wizard

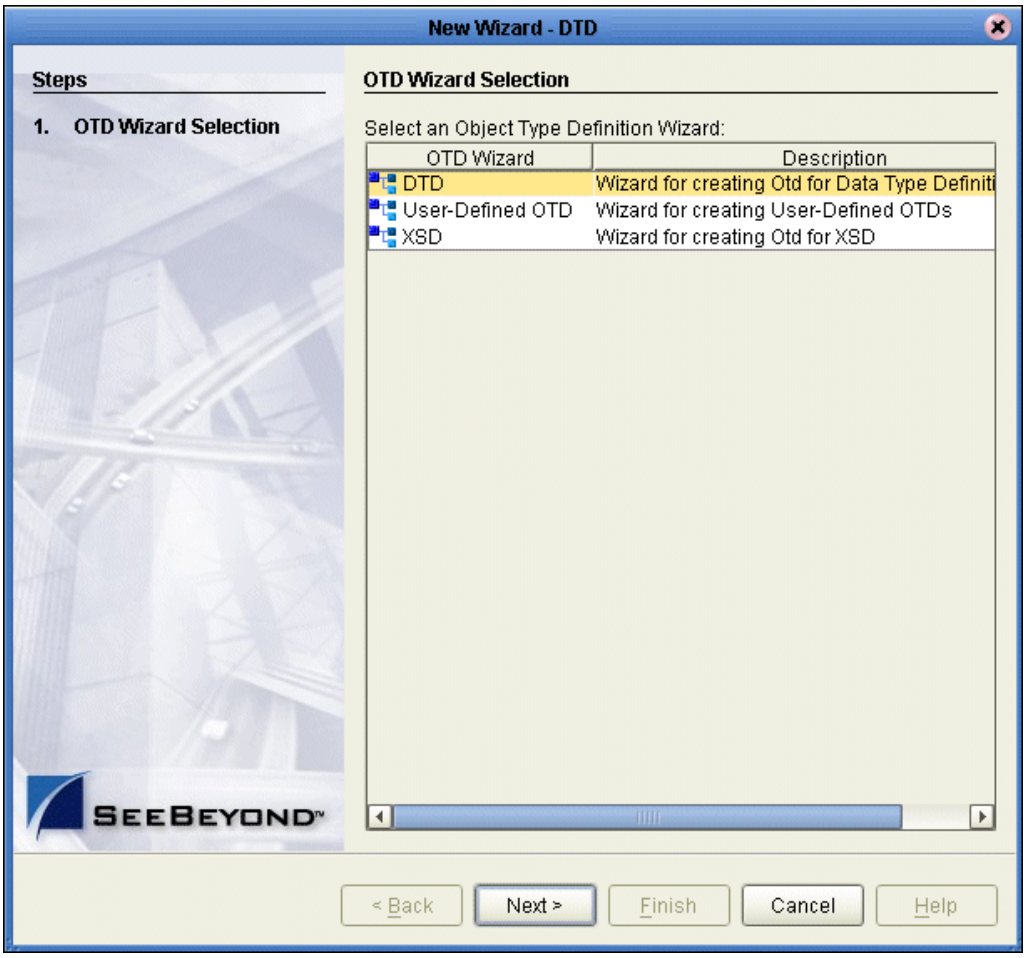
You must create a Data Type Definition (DTD) OTD as an input file for this HTTP(S) sample Project. You create OTDs by using the OTD Wizard.

To create a new DTD using the OTD Wizard

1 In the **Enterprise Explorer**, right-click **HTTPS_Client_Project_BPEL** and select **New > Object Type Definition** from the pop-up menu.

The OTD Wizard Selection window appears. See Figure 9.

Figure 9 OTD Wizard Selection

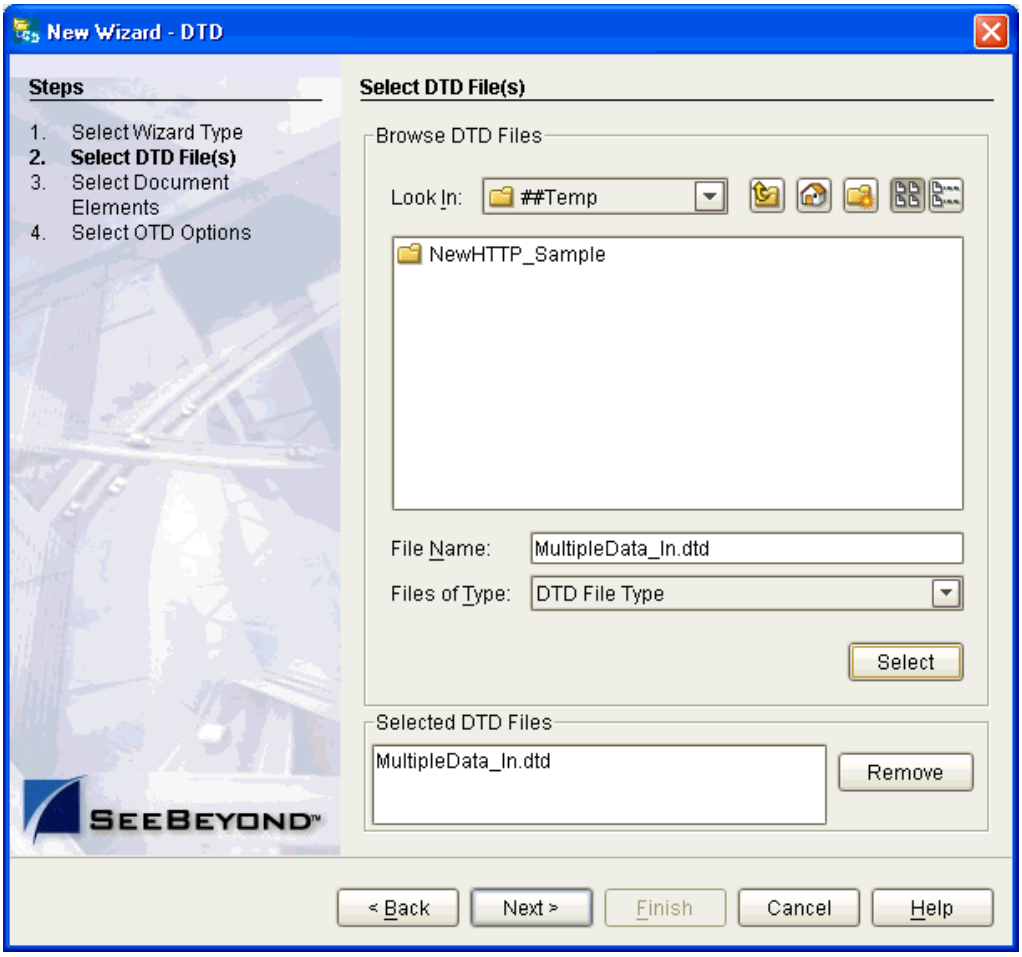


2 From the OTD Wizard Selection window, select **DTD** from the OTD Wizard column. Click **Next**.

3 From the Include DTDs to Selected List window, browse to the **MultipleData_In.dtd** located in the sample folder. Click **Select**.

4 The **MultipleData_In.dtd** file appears in the **List of Selected DTDs** pane. See **Figure 10 on page 44**.

Figure 10 Include DTDs to Selected List



5 Click **Next**.

6 From the Select Document Elements window, select **MultipleData_In_with_top_website** and click **Finish**.

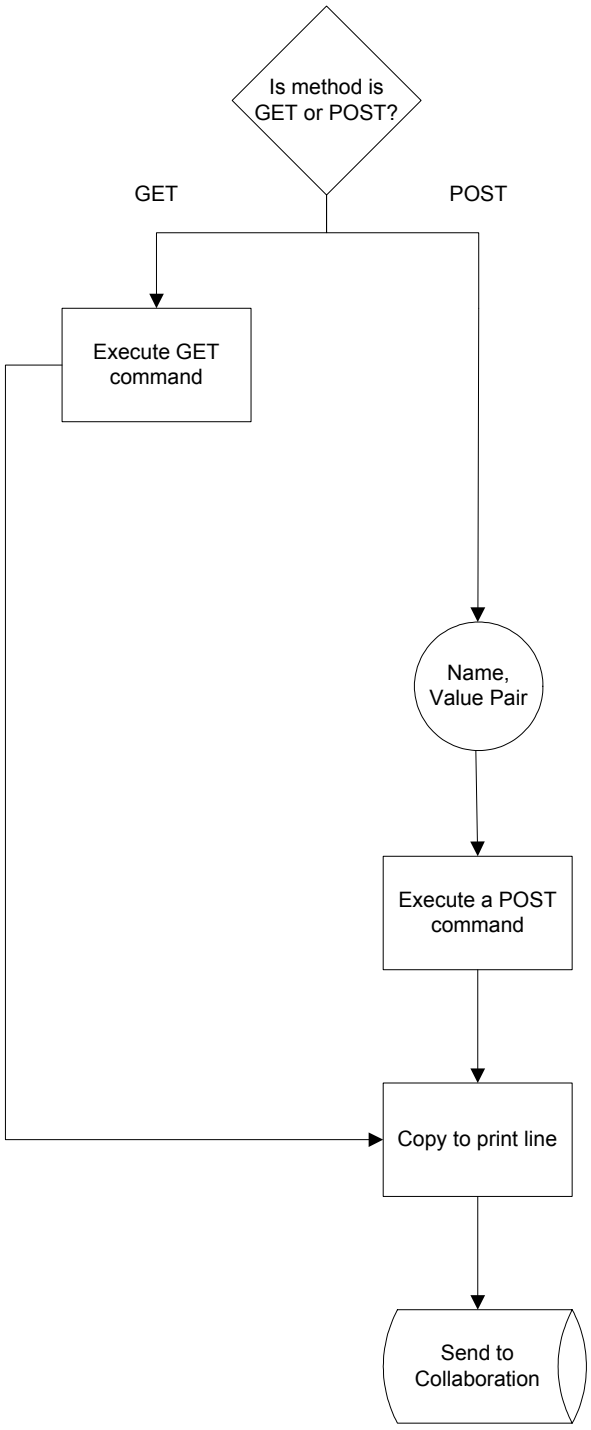
A Message dialog box appears if the OTD is successfully created. The OTD appears in the **Project Explorer** as the OTD icon **MultipleData_In_with_top_website**.

4.4.6 Creating a Business Process

Once you have designed your Business Process for this sample, you can use the eGate Enterprise Designer to create it. See **"Reviewing the Business Process Project: Client" on page 38** for a description of the sample's Business Process.

The logic of the Business Process is shown in **Figure 11 on page 45**.

Figure 11 Logic of the Business Process



This scenario sets up two possible decisions, called Cases in eInsight. If the inbound file requests a GET operation, it is routed to Case 1. If the inbound file requests a POST operation, it is routed to Case 2. Table 3 shows how these cases operate of this Business Process.

Table 3 Business Process Cases

Case	Activity	Result
Case 1: GET operation	Requests that the Business Process get information from the HTTP server.	Appropriate information is retrieved.
Case 2: POST operation	Requests that the Business Process posts information to the HTTP server.	Appropriate information is posted.

To create a Business Process

1. Right-click the name of the sample Project, **HTTPS_Client_Project_BPEL**, in the **Project Explorer** and choose **New > Business Process** from the pop-up menus. You can use the default name, **HTTP_CLIENT_BP**.

   A blank Business Process canvas appears in the right pane, along with the Business Process toolbar.

2. In the **Project Explorer** expand the icons for **SeeBeyond > eWays** for **File** and **HTTP**. Also expand the icon for the **MultipleData_In_with_top_website** OTD.

3. Arrange the **Start** and **End** icons at opposite sides of the canvas, then drag the following icons onto the canvas:

   From the **Project Explorer**:

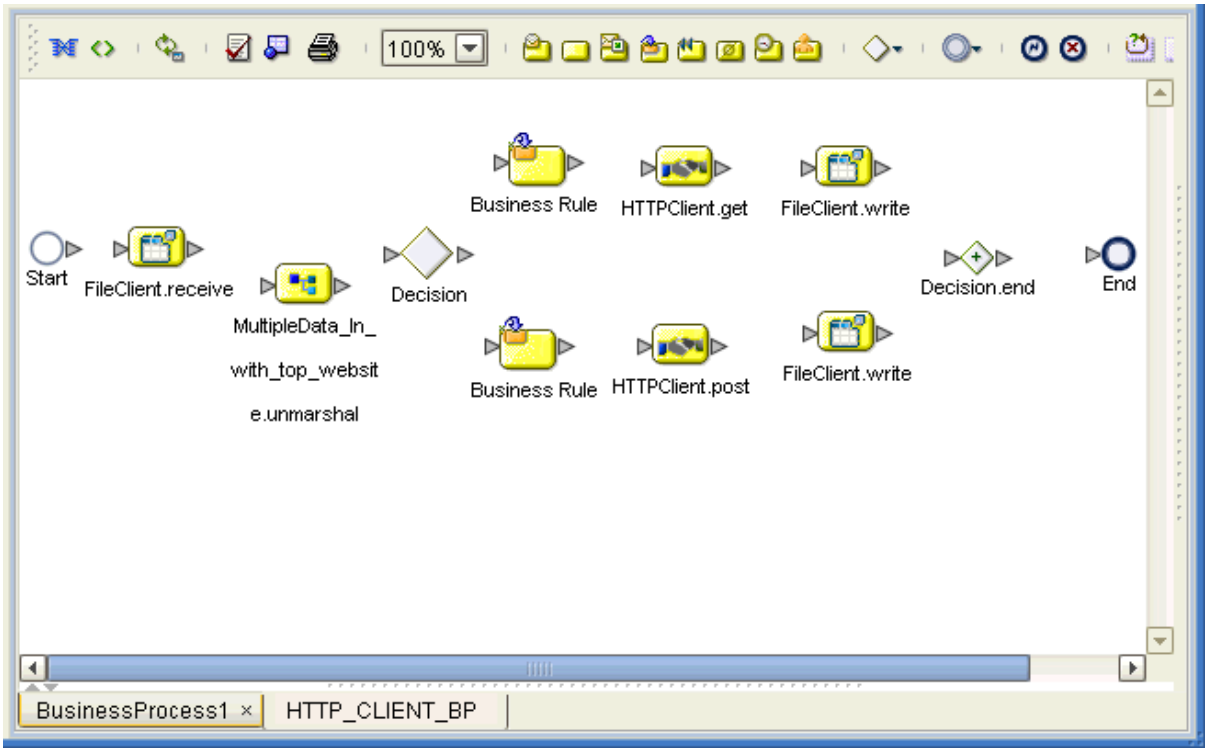
   - **HTTP eWay** (server)
     - One **get** icon
     - One **post** icon
   - **File eWay**
     - One **receive** icon
     - Two **write** icons
   - **MultipleData_In_with_top_website.unmarshal** OTD icon

   From the Business Process canvas toolbar:

   - **Decision** (a **Decision End** icon also appears)
   - Two **Business Rule** icons, for your two cases

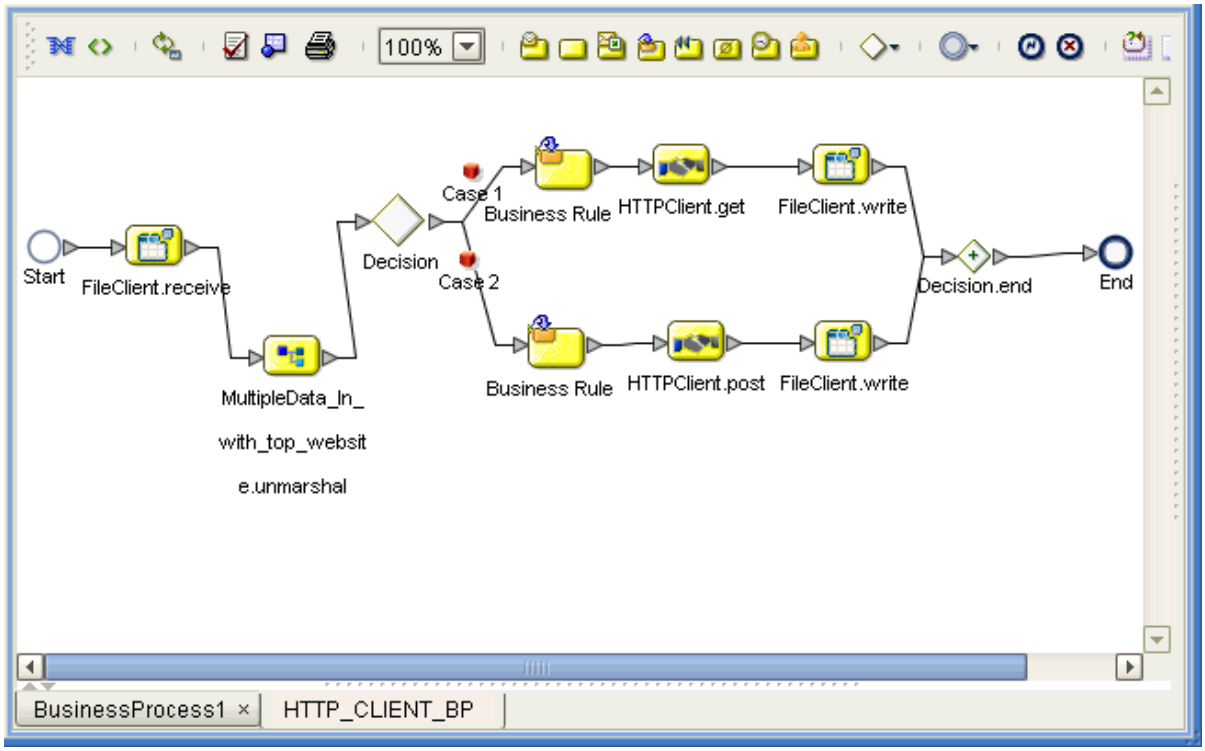
4 Again by dragging, arrange these icons on the canvas as shown in Figure 12.

Figure 12 Business Process Icons: Client



5 By dragging from one icon to another, link the icons on the canvas, as shown in Figure 13.

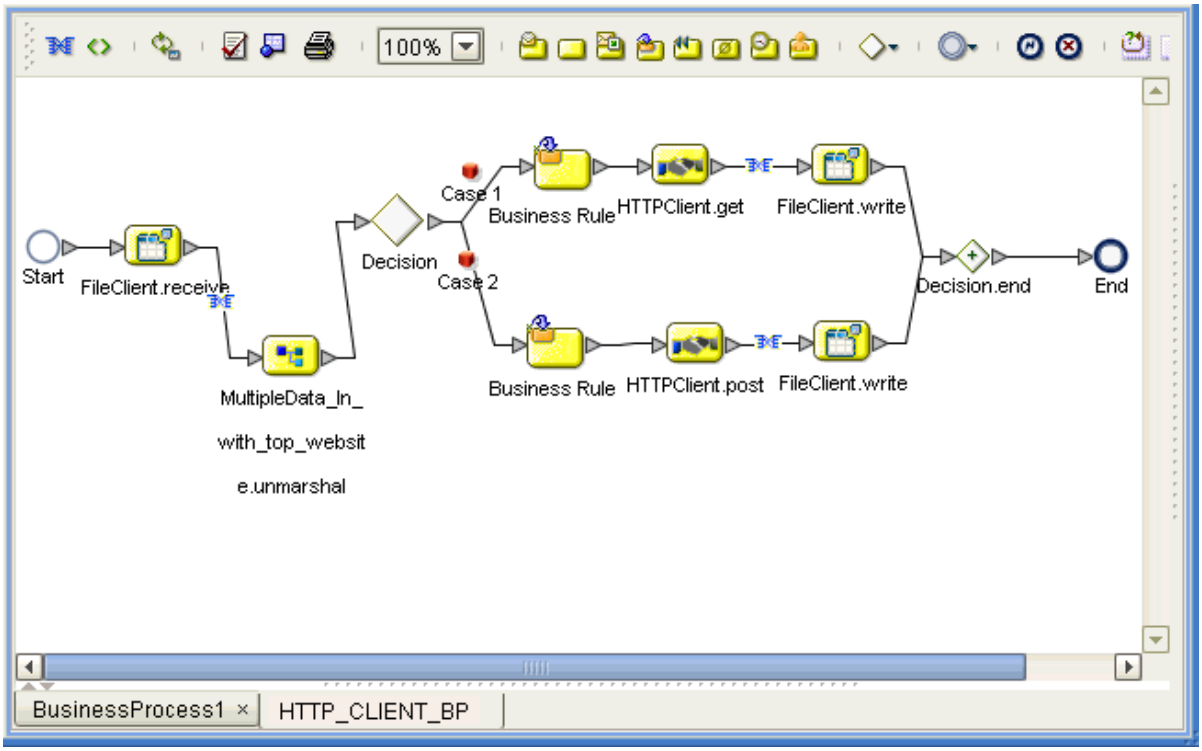
Figure 13 Business Process With Links: Client



Two **Case** icons appear between the **Decision Gate** and each of your **Business Rule** icons.

6 You must add additional Link Business Rules (represented by a small blue, star-shaped icons) to the appropriate links, as shown in Figure 14. To do this operation, right-click on the desired link and choose **Add Business Rule** from the pop-up menu. See Figure 14 for the appropriate links where you must add these Link Business Rules.

Figure 14 Business Process With Link Business Rules: Client



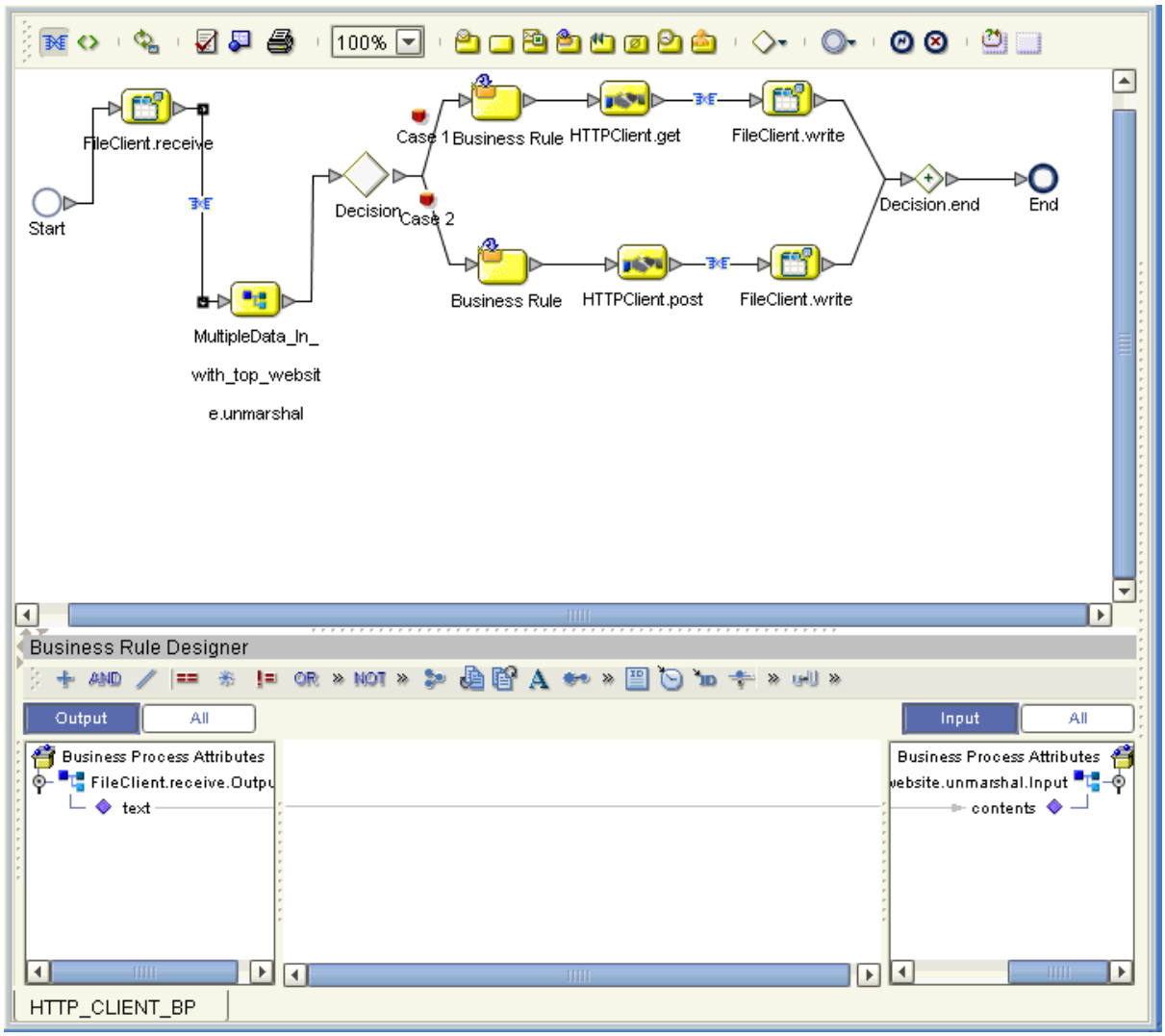
For each Business Rule (Link and **Business Rule** icon), you must create the settings you want in the Business Rule Designer.

7 Select the Link Business Rule on the left, then click the **Map Business Process Attributes** icon in the toolbar.

The Business Rule Designer pane appears at the bottom of the window. Use the Business Rule Designer to create your Business Rules.

8 Set properties For this Link Business Rule by dragging the **text** node from the **Output** pane, and dropping it onto the **contents** node you want to assign it to, in the **Input** pane. In this way, create the first Link Business Rule, as shown in Figure 15.

Figure 15 Business Rule Designer: First Link Business Rule



9 In the same way as you did previously, create additional Link Business Rules, as shown in Figure 16 and **Figure 17 on page 52**.

Figure 16 Business Rule Designer: Second Link Business Rule

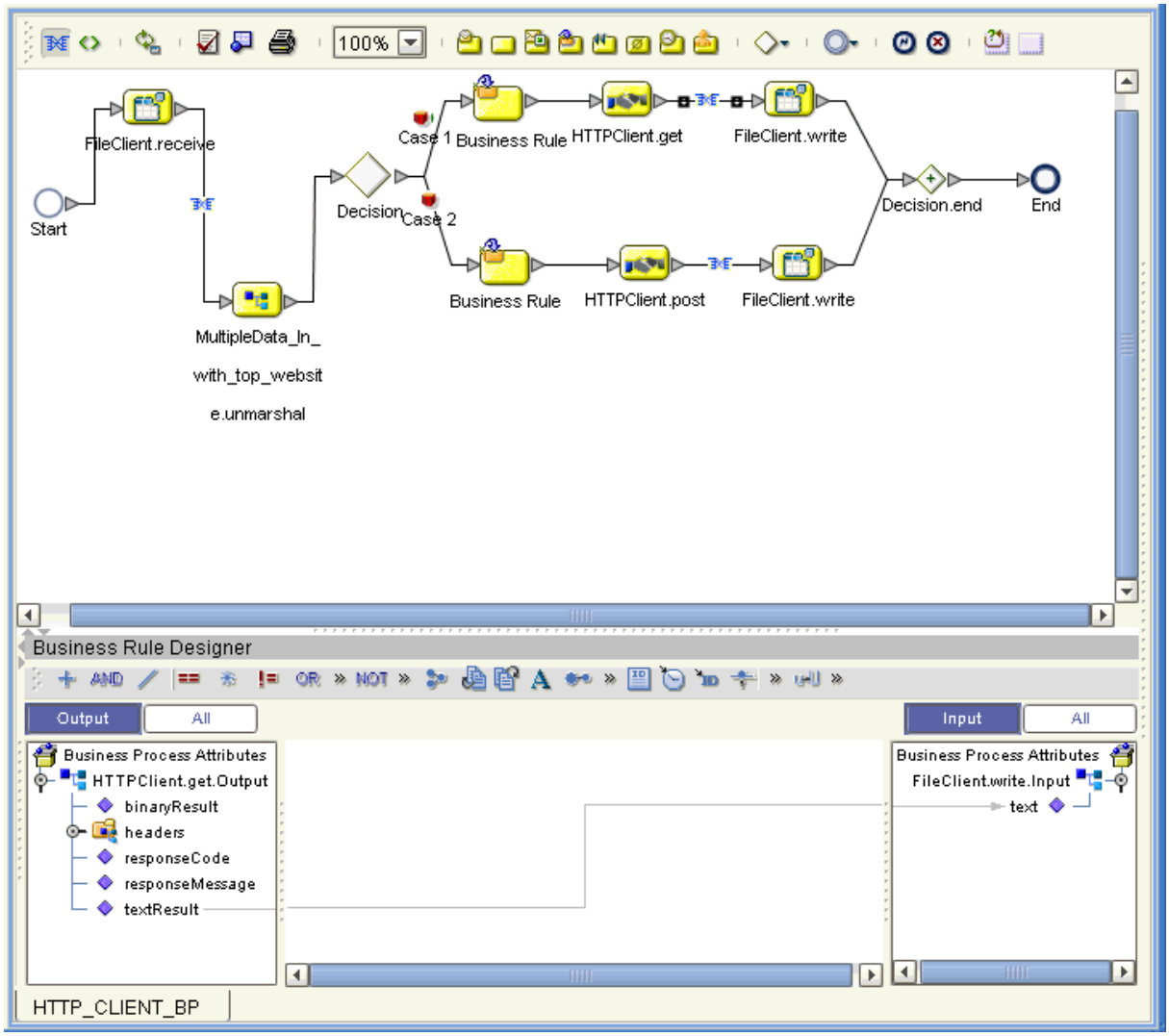
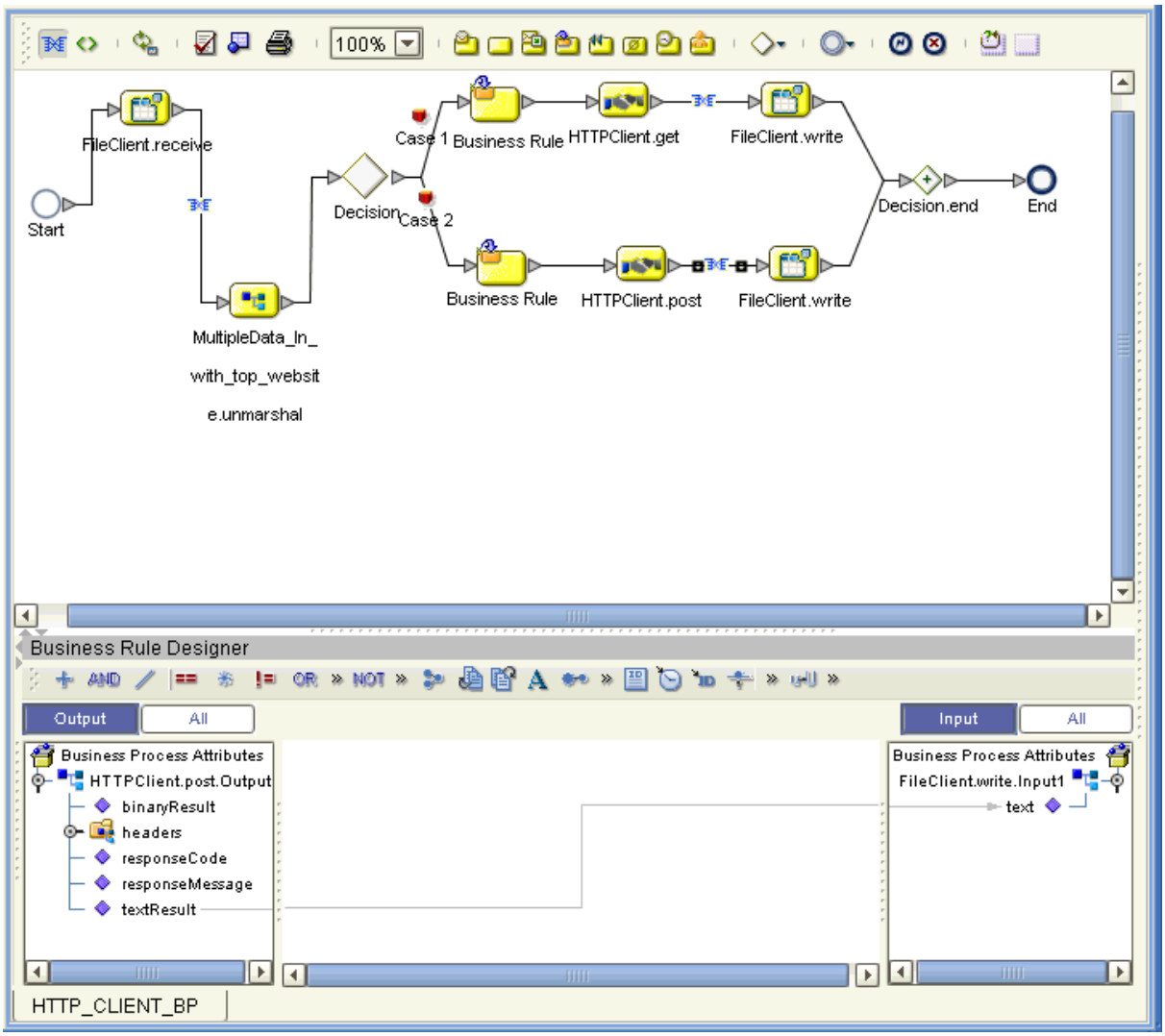


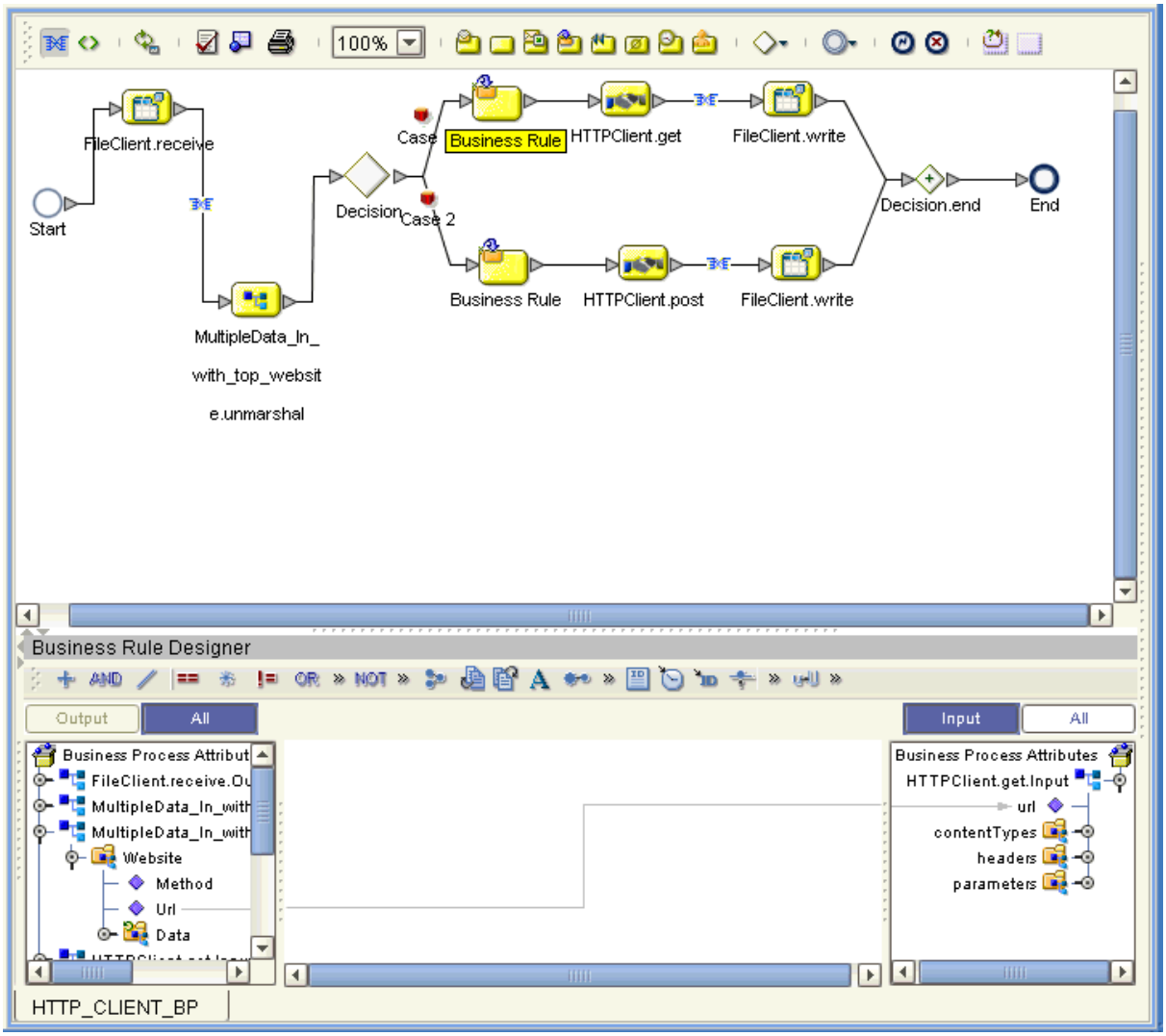
Figure 17 Business Rule Designer: Third Link Business Rule



10 In addition, you must set properties for the **Business Rule** icon components. Select the desired **Business Rule** icon component to open the Business Rule Designer (Figure 18).

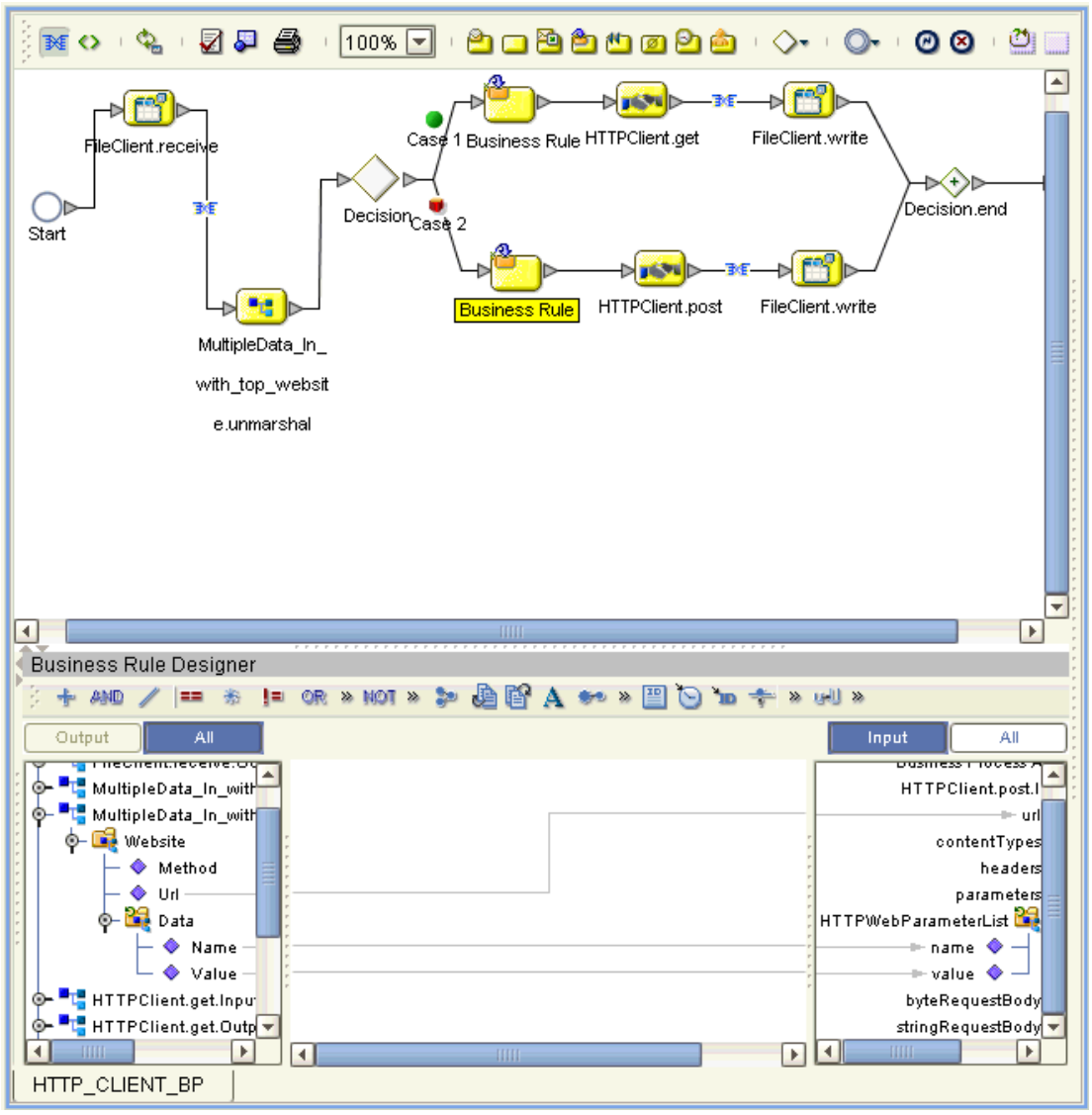
Using the Business Rule Designer in the same way as you did previously, set properties for the **Business Rule** icon component for Case 1 by dragging and dropping the nodes, as shown in Figure 18.

Figure 18 Business Rule Designer: Case 1 Business Rule



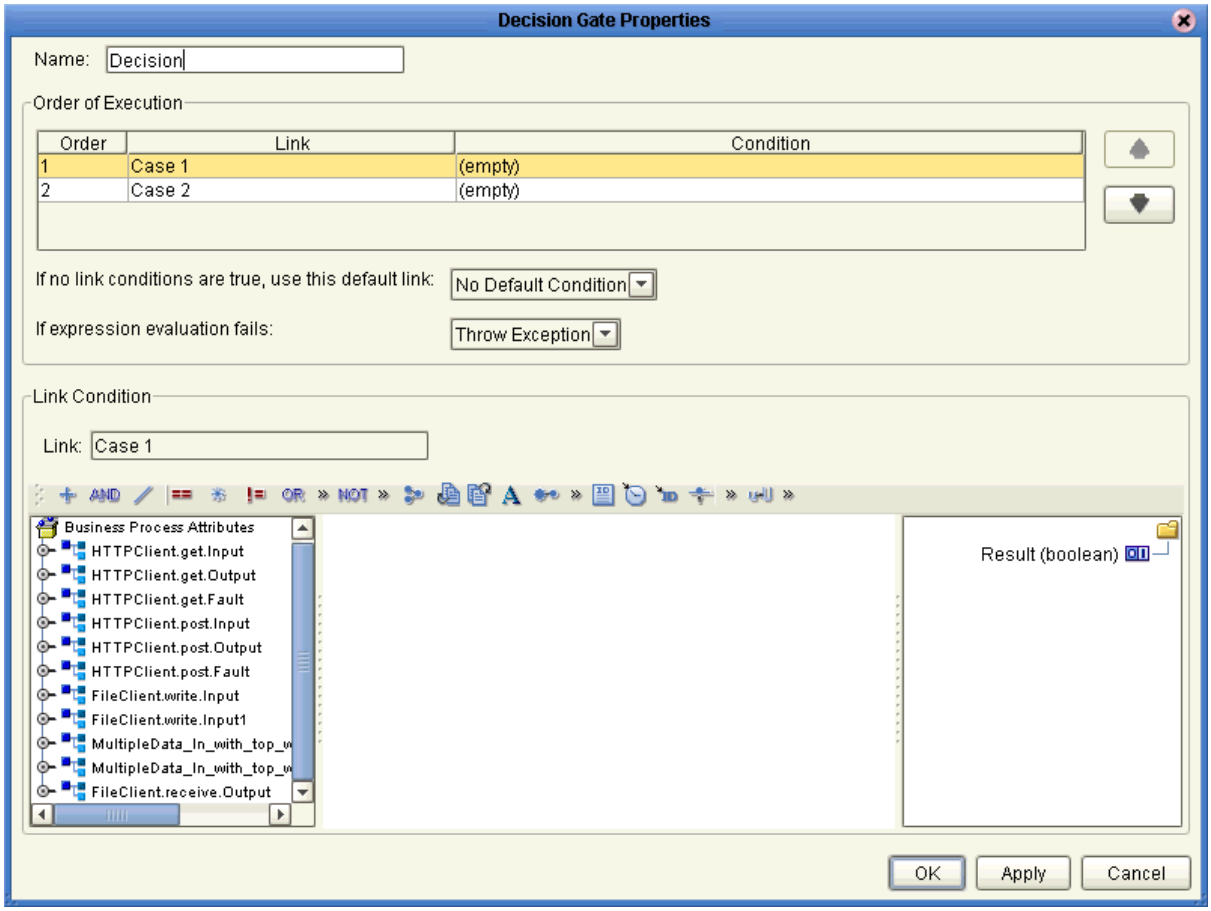
11 Set properties For the **Business Rule** icon component for Case 2 by dragging and dropping the nodes, as shown in Figure 19.

Figure 19 Business Rule Designer: Case 2 Business Rule



12 Double-click the **Case 1** (red) icon to set the **Decision Gate** properties for the cases. The **Decision Gate Properties** dialog box opens. See Figure 20.

Figure 20 Decision Gate Properties Dialog Box



13 For **Case 1**, add a **string literal** by dragging the icon from the toolbar. Call the literal **GET**.

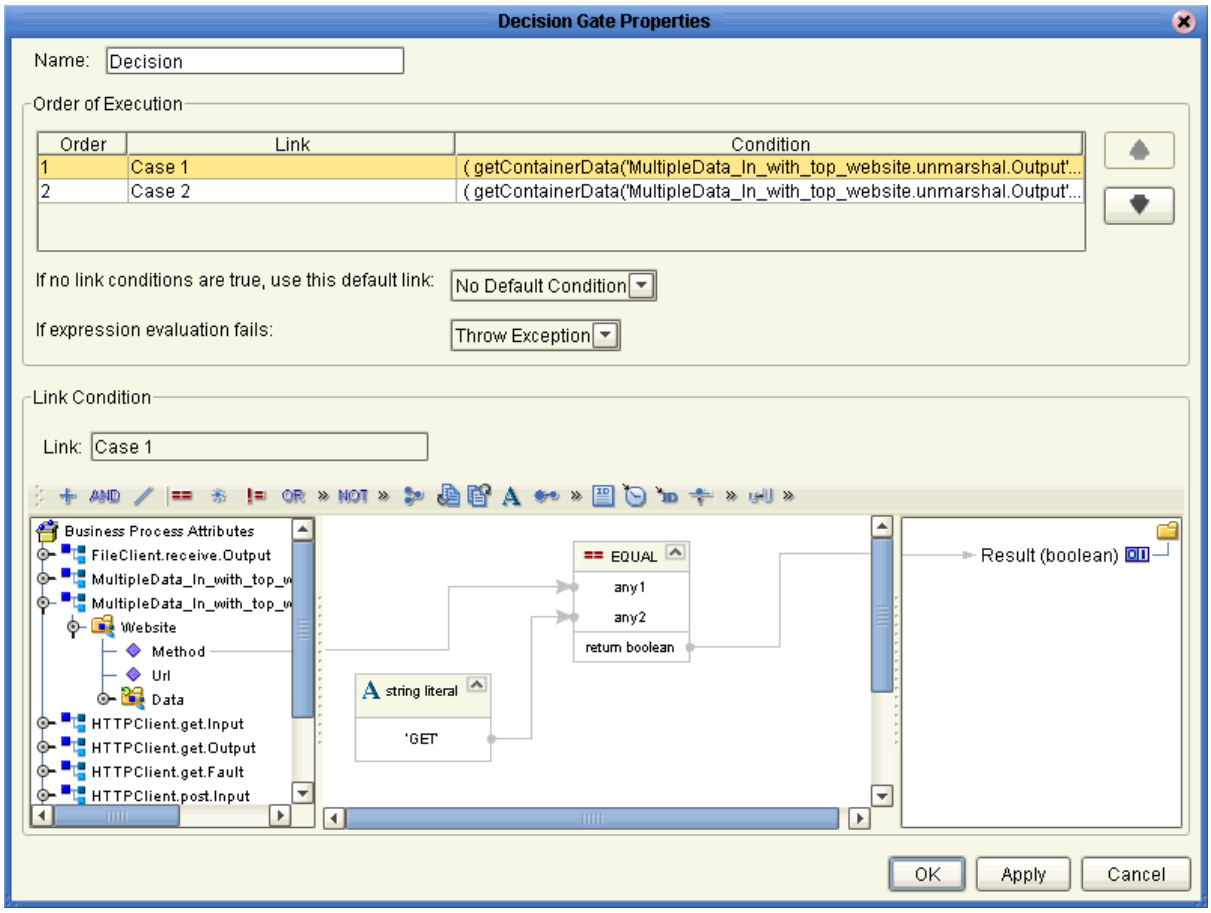
14 By dragging the icon from the toolbar, add an **EQUAL**.

15 Drag Method under **MultipleData_In_with_top_website.unmarshal.Output** to **any1** under **EQUAL** in the left pane.

16 Drag **GET** under **string literal** to **any2**.

17 Drag **return boolean** under **EQUAL** to **Result (boolean)** in the right pane. See Figure 21.

Figure 21 Decision Gate Properties Dialog Box: Case 1



18 For **Case 2**, add a **string literal** by dragging the icon from the toolbar. Call the literal **POST**.

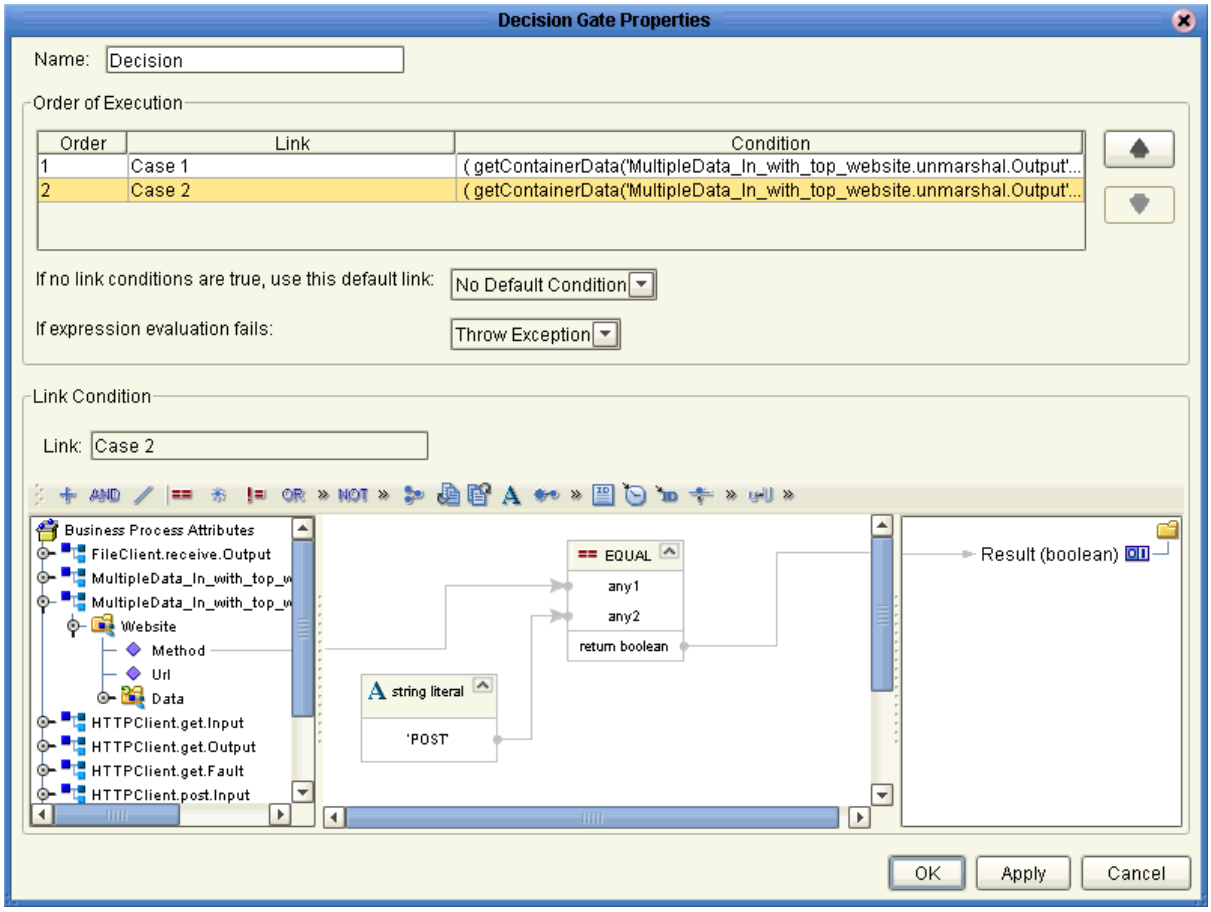
19 By dragging the icon from the toolbar, add an **EQUAL**.

20 Drag Method under **MultipleData_In_with_top_website.unmarshal.Output** to **any1** under **EQUAL** in the left pane.

21 Drag **POST** under **string literal** to **any2**.

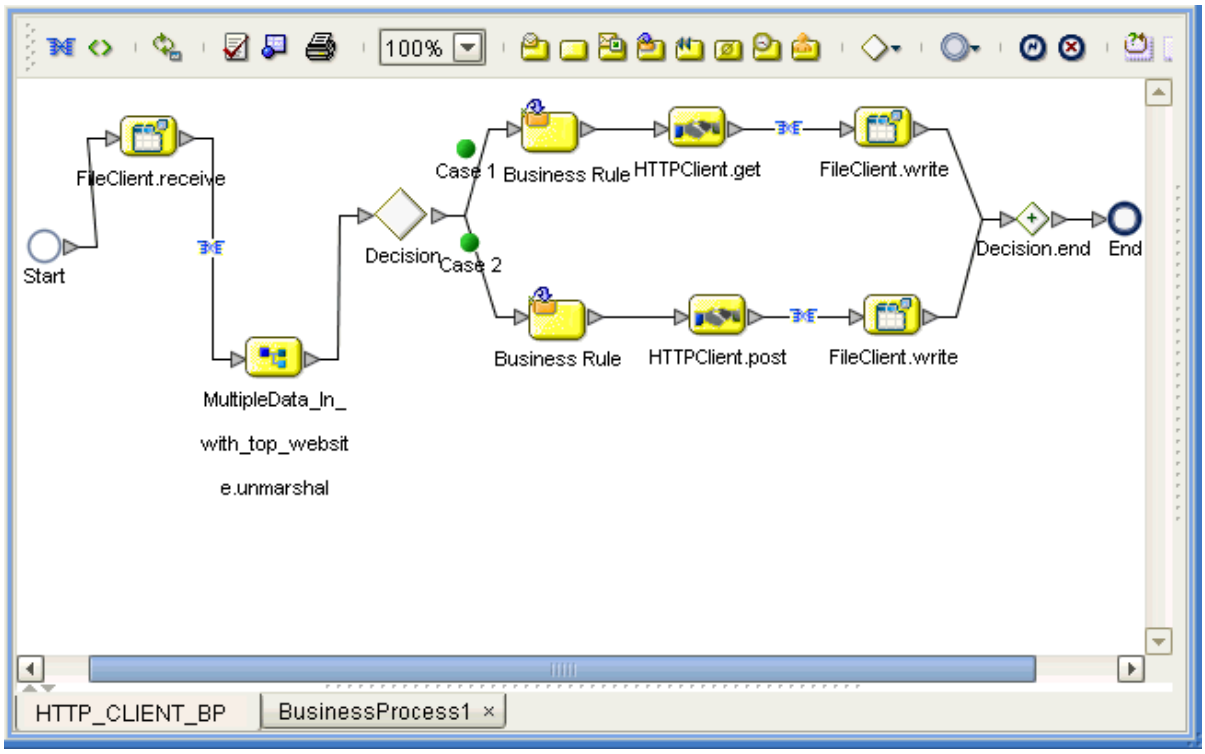
22 Drag **return boolean** under **EQUAL** to **Result (boolean)** in the right pane. See Figure 22.

Figure 22 Decision Gate Properties Dialog Box: Case 2



23 Figure 23 shows your finished Business Process.

Figure 23 Finished Business Process: Client



24 Click **Save** on the Enterprise Designer toolbar to save your Business Process.

After you have finished creating your Business Process, you can use it to define one or more of the eGate Services on your Connectivity Map.

4.4.7 Creating a Connectivity Map

A Connectivity Map provides a canvas for assembling and setting properties for a Project's components.

To create a Connectivity Map

1 In the eGate Enterprise Designer's **Project Explorer** pane, right-click the new Project's name and select **New** then **Connectivity Map** on the pop-up menus.

2 The new Connectivity Map appears and adds a node for the Connectivity Map under the Project on the **Project Explorer** tree labeled **CMap1**.

3 Rename the new Connectivity Map **HTTP_CLIENT_BPEL_CM**.

The icons in the toolbar represent the components used to populate the Connectivity Map workspace. When linked together, these components define the Connectivity Map for your Project.

4.4.8 Selecting External Applications

When creating a Connectivity Map, you can associate any Service, in this case a Business Process, with an external application. For example, to establish a connection to HTTP, you must first select **HTTP** as the external application to use in your Connectivity Map.

To select external applications

1. Click the **External Application** icon on the Connectivity Map toolbar.
2. Select the external applications necessary for your Project. For this sample, select the **File** and **HTTP Client** external applications. Icons representing these external applications are then added to the Connectivity Map toolbar.

4.4.9 Populating the Connectivity Map

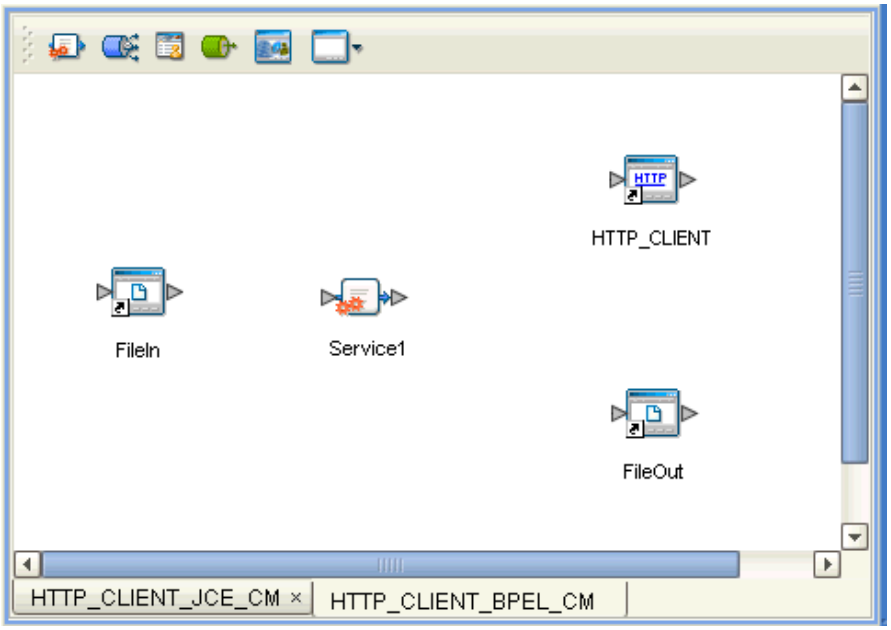
Add the Project components to the **HTTP_CLIENT_BPEL_CM** Connectivity Map by dragging the icons from the toolbar to the canvas. This operation creates the components for you.

For this sample Project, drag and drop the following components onto the Connectivity Map canvas.

- Two File eWays/external applications
- One Service (rename to **HttpBpelService**)
- HTTP eWay/client external application

Figure 24 shows the components on the Connectivity Map.

Figure 24 Connectivity Map With Components: HTTPS_Client_Project_BPEL



Rename the **Service1** component to **HttpBpelService**. Name the other components as shown in the previous figure. Be sure to save the new Connectivity Map before you proceed. You can click **Save** on the Enterprise Designer toolbar for this purpose.

4.4.10 Defining the Business Process

After you have created the Connectivity Map, you need to define your Business Process, that is, combine the **Business Process** icon with the **Service** icon in the Connectivity Map.

To do this operation, you can drag and drop the **HTTP_CLIENT_BP** icon from the **Project Explorer** tree onto the desired **Service** icon in the Connectivity Map.

If the Collaboration is successfully defined, the "gears" icon changes from red to yellow.

Note: *You can also drag the desired Business Process onto the Connectivity Map by itself and create a Business Process Service without first creating the Service component.*

4.4.11 Binding OTDs in Business Processes

After you have set up the Business Process, you need to bind, that is, associate the appropriate OTDs with the desired components. For details on this operation, see the *eGate Integrator User's Guide*.

4.4.12 Creating the Project's Environment

This section provides general procedures for creating an Environment for your Project. For a complete explanation of this operation, see the *eGate Integrator User's Guide.*

To create an Environment

1. From the Enterprise Designer, click the **Environment Explorer** tab on the Enterprise Explorer.
2. Under the current **Repository** icon in the **Environment Explorer**, create a new environment for your Project and name it as desired.
3. In the **Environment Explorer**, right-click the **Environment** icon and select the desired external systems from the pop-up menu. Include external systems for the HTTP(S) eWay (**HTTP External System**) and the File eWays.
4. Use the same pop-up menu to create a Logical Host for your Project, and name it as desired.
5. Click **Save** and return to the **Project Explorer** tab.

4.4.13 Setting eWay Properties

You must set the eWay properties for your specific system and for the current Project, using the eGate Enterprise Designer. For directions on accessing and using the eWay **Properties** dialog box, as well as a complete explanation of the HTTP(S) eWay properties, see **Chapter 3**.

Note: *See* **"Security/SSL" on page 24** *for instructions on how to set SSL-related properties.*

To set properties for the File eWays

1. From the **Project Explorer**, open the **HTTP_CLIENT_BPEL_CM** Connectivity Map for the sample Project.
2. To change the default properties for the inbound File eWay, click the **FileIn** external system's eWay icon. For this eWay, select the **Inbound** properties.

   The eWay **Properties** dialog box appears. This eWay can use the current default properties settings for the sample Project. However, if you are using different file names and/or folders than the defaults, you must enter the names of the files/folders you are using.
3. Click **OK** to save the settings and close the eWay **Properties** dialog box.
4. To change the default properties for the outbound File eWay, click the **FileOut** external system's eWay icon. For this eWay, select the **Outbound** properties.

   Use your file and folder names if they differ from the defaults. For all other properties, you can use the defaults.

   In addition, for this project, use the output file name **output%d.html** for that properties.
5. Click **OK** to close the window and save.

To set properties for the HTTP(S) eWay

1. To begin changing the default properties for the HTTP(S) eWay, click the **HTTP_CLIENT** external system's eWay icon on the Connectivity Map.

   The eWay **Properties** dialog box appears.
2. From the upper left pane of the eWay **Properties** dialog box, select the **properties** folder, then the desired subfolders.

   The properties settings appear in the **Properties** pane on the left.
3. Use the default settings for all properties. Click **OK** to close the window and save.
4. From the Enterprise Designer, click the **Environment Explorer** tab.
5. Create a new environment for your project then create a **HTTP** external system within that environment (**HTTP_CLIENT**) to correspond to the system you created on the Connectivity Map in the **Project Explorer**. For details on how to do these operations, see the *eGate Integrator User's Guide.*
6. In the left pane, right-click the **HTTP_CLIENT** external system icon and select **Properties** from the pop-up menu.

The eWay **Properties** dialog box appears. You can use this window to set additional properties settings, in the same way as you did for the **Project Explorer** properties. For this Project you can use the defaults.

7 Click **OK** to close the window and save.

Project deployment

For information on how to deploy your Project, see **"Deploying a Project" on page 74**.

4.4.14 Alerting and Logging

eGate provides an alerting and logging feature. This operation allows the monitoring of messages and captures any adverse messages in the order of severity, based on a configured severity level and higher. For details on how to enable logging and alerting, see the *eGate Integrator User's Guide*.

Note: *The alerts/status notifications for the HTTP(S) eWay in client mode are currently limited to* ***Started****,* ***Running****,* ***Stopping****, and* ***Stopped****.*

4.5 Building the Business Process Project: Server

This section explains how to implement the HTTP(S) eWay using the eGate Project server sample with an eInsight Business Process, which is included on your installation CD-ROM. The sample is named **HTTP_Server_Project_BPEL**. It allows you to observe a data-exchange scenario involving eGate and the HTTP(S) eWay.

This section also explains how to implement this sample Project, including the HTTP(S) eWay in server mode. You can also use the procedures given in this chapter to create your own Projects based on the sample provided.

Importing the Project

You can import this sample Project using the procedure under **"Importing Sample Projects" on page 38**.

The path location of this sample file is the same as that given under the previous procedure, and:

- The container file name you are looking for is **HTTPS_eWay_Sample.zip**.
- The Project file name is **HTTP_ServerSample_BPEL.zip**.
- Be sure to name the Project **HTTPS_Server_Project**.

4.5.1 Server Sample Project: Overview

The server HTTP(S) eWay sample Project with an eInsight Business Process demonstrates how the HTTP(S) eWay can receive information via HTTP, from a server.

Before Running the Project

Before you can run the Project, you must first copy the following .**html** input form file into any directory:

- **postHTTPS**

The content of **postHTTPS.html** is:

```
<HTML><HEAD><TITLE>HTTPS Test Page</TITLE></HEAD>
<BODY>
<FORM ACTION="http://localhost:18004/HttpServer_DP1_servlet/
    HttpServerSample" METHOD=POST>
<TABLE>
<TR><TD>First Name:</TD><TD><INPUT NAME=fname></TD></TR>
<TR><TD>Last Name:</TD><TD><INPUT NAME=lname></TD></TR>
<TR><TD>EMail:</TD><TD><INPUT NAME=email></TD></TR>
<TR><TD>Sex:</TD><TD><INPUT type="radio" name="sex"
    value="Male">Male</TD></TR>
<TR><TD></TD><TD><INPUT type="radio" name="sex"
    value="Female">Female</TD></TR>
<TR><TD></TD><TD></TD></TR>
</TABLE>
<BR>
<CENTER><INPUT TYPE=submit VALUE="Submit"></CENTER>
</FORM>
</BODY>
</HTML>
```

You must make a change in the HTML code shown previously. In the code where it shows:

**<FORM ACTION="http://localhost:18004/HttpServer_DP1_servlet/
HttpServerSample" METHOD=POST>**

You must make changes based on your own Environment. The logic for the ACTION parameter is:

**http://<IS Server Name>:<IS port>/<Deployment_name>_servlet/
<servlet_url from properties>**

Project Operation

Figure 25 shows the original form.

Figure 25 Server Sample Project: Original Form

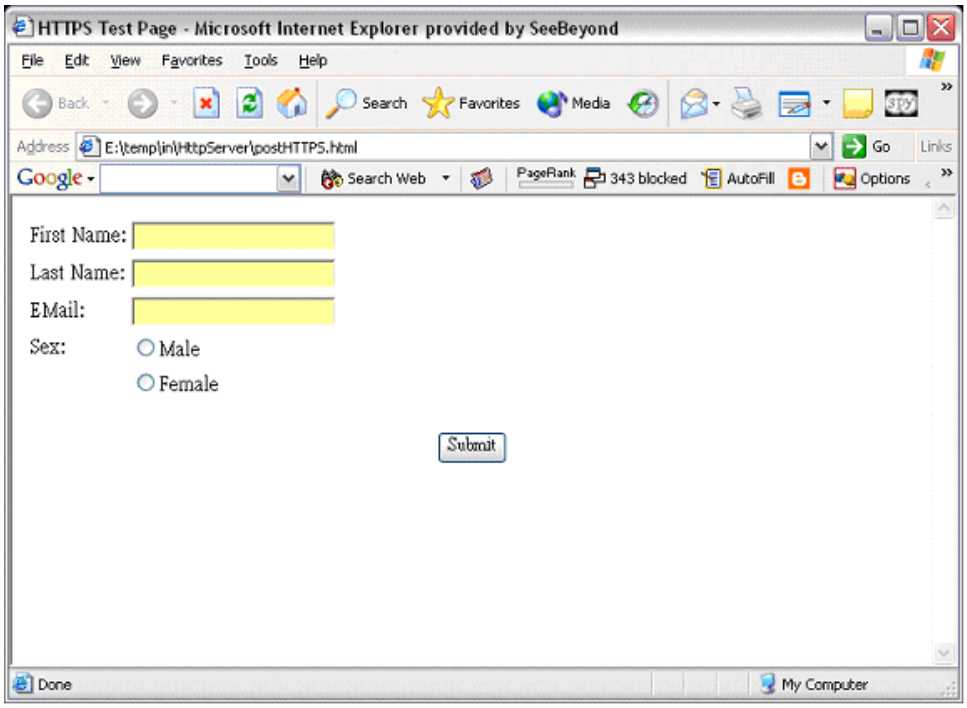


Figure 26 shows the input form.

Figure 26 Server Sample Project: Input Form

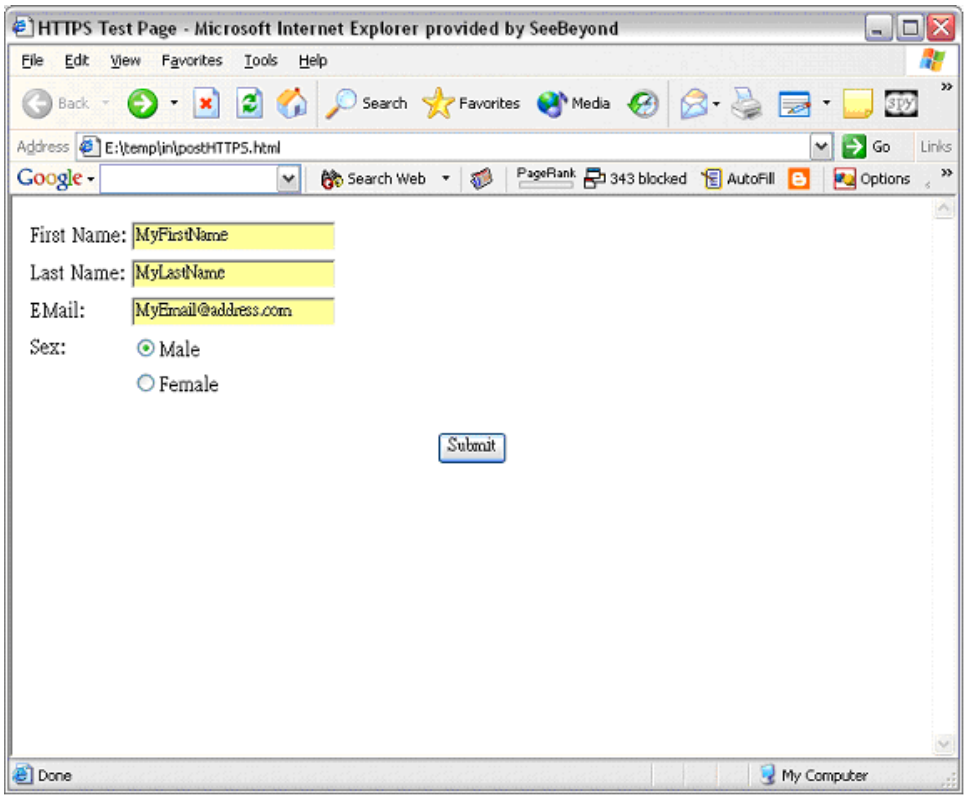
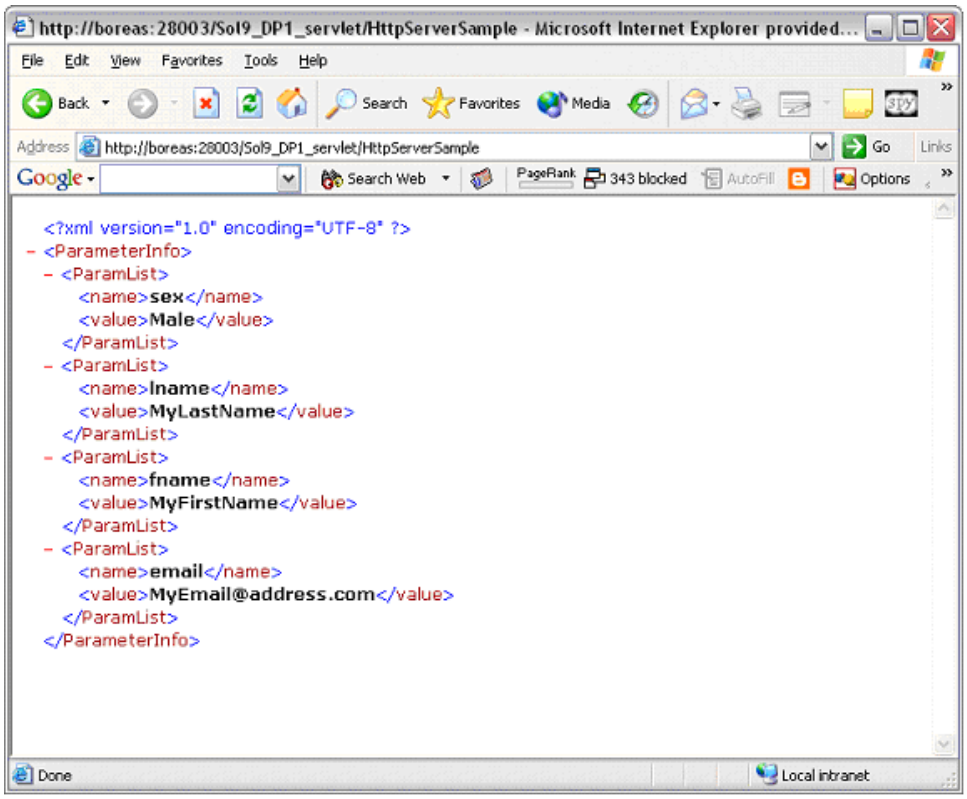


Figure 27 shows the output form.

Figure 27 Server Sample Project: Output Form



The input for the Project is a name/value pair, and it returns the entire list of parameters. A DTD file (**HTTPS_ParamList.dtd**) is used to marshal the list, so you must use the DTD wizard to convert this file to an eGate OTD.

To import the sample Project

- For instructions on how to import the sample Project see **"Importing Sample Projects" on page 38**.
- You are looking for the container file **HTTPS_eWay_Sample.zip**. The file name of the sample file is **HTTPS_ServerSample_BPEL.zip**.

4.5.2 Server Sample Project Components and Operation

The HTTP(S) eWay server sample Project demonstrates how the HTTP(S) eWay processes information via an eInsight Business Process.

Project Components

The server Project has the following components:

- HTTP(S) eWay
- HTTP external application: **HTTPServer1**
- Business Process processing data: **HTTPS_BP1** (created from the **Service** icon)

Project Operation

The server Project operates as follows:

- **HTTPServer1**: The HTTP server external application or system; the HTTP(S) eWay handles inbound communication with this system.
- **HTTPS_BP**: Receives instructions from the HTTP server external application via the HTTP(S) eWay.

4.5.3 Creating Project, Connectivity Map, and External Application

Follow the same steps as those provided under the following sections:

- **"Creating a New Project" on page 41**
- **"Creating a Connectivity Map" on page 58**
- **"Selecting External Applications" on page 59**

Name the components as follows:

- **Project**: **HTTP_Server_Project_BPEL**
- **Connectivity Map**: **HTTPS_CM**

Select the following external application:

- **HTTP Server**

4.5.4 Creating the OTD

An OTD contains a set of rules that define an object, which encodes data as it travels through eGate. OTDs are used as the basis for creating Business Processes for a Project. The HTTP(S) eWay provides HTTP OTD for this purpose. See **Chapter 6** for complete information on how to use this OTD.

DTD OTD Wizard

You must create a Data Type Definition (DTD) OTD as an input file for this HTTP(S) sample Project. You create OTDs by using the OTD Wizard. For details on how to convert the **HTTPS_ParamList.dtd** file into an eGate OTD, see **"Using OTDs" on page 42**.

Name the new OTD **HTTPS_ParamList_ParameterInfo**.

4.5.5 Creating a Business Process

Once you have designed your Business Process for this sample, you can use the eGate Enterprise Designer to create it. See **"Server Sample Project: Overview" on page 62** for a description of the sample's Business Process.

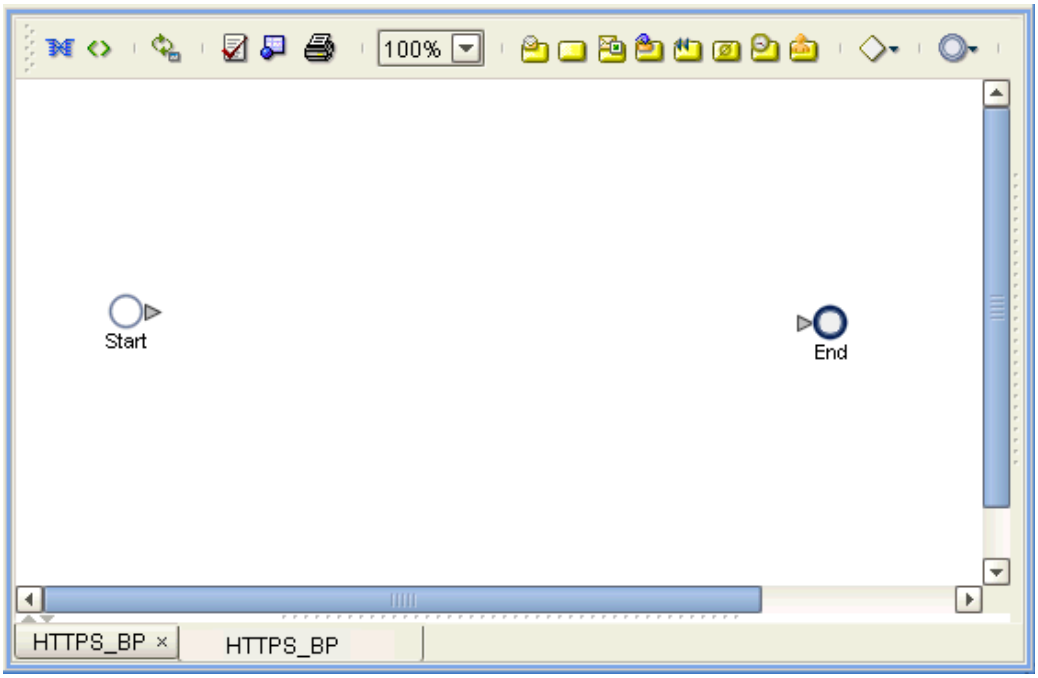
To create a Business Process

1. Right-click the name of the sample Project, **HTTPS_Server_Project**, in the **Project Explorer** and choose **New > Business Process** from the pop-up menus. You can use the name, **HTTPS_BP**.

   A blank Business Process canvas appears in the right pane, along with the Business Process toolbar.

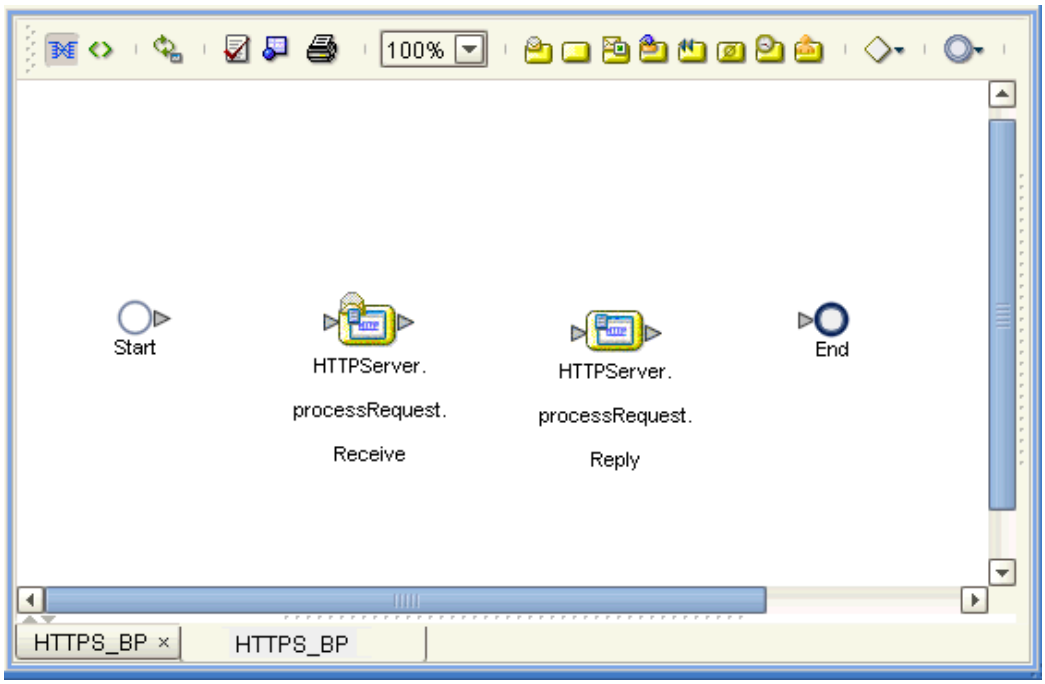
2. In the **Project Explorer**, expand the icons for **SeeBeyond > eWays > HTTPServer**.
3. Arrange the **Start** and **End** icons at opposite sides of the canvas, as shown in Figure 28.

Figure 28 Business Process Icons: Server



4 From the Project Explorer pane, drag the **processRequest** icon under the HTTPServer OTD nodes onto the canvas between the **Start** and **End**. See Figure 30.

Figure 29 Business Process Icons for Receive and Reply

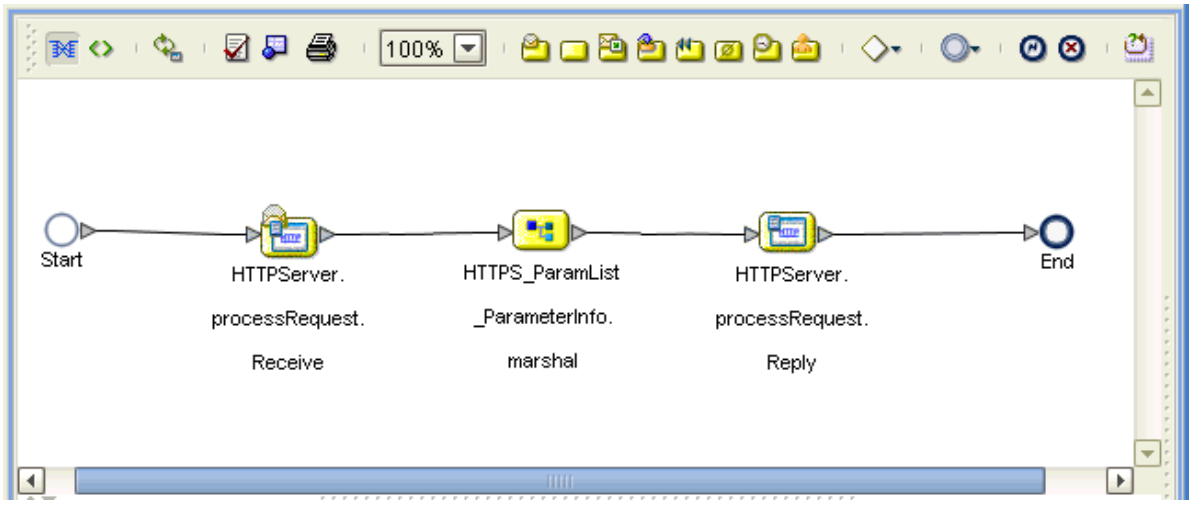


The single icon becomes two, as shown in Figure 30. If the icons appear out of line, drag them until the icons appear, as shown in Figure 30.

5 From the Project Explorer pane, drag the **HTTPS_ParamList_ParameterInfo** OTD's **marshal** operation onto the canvas between the two **HTTPServer** icons.

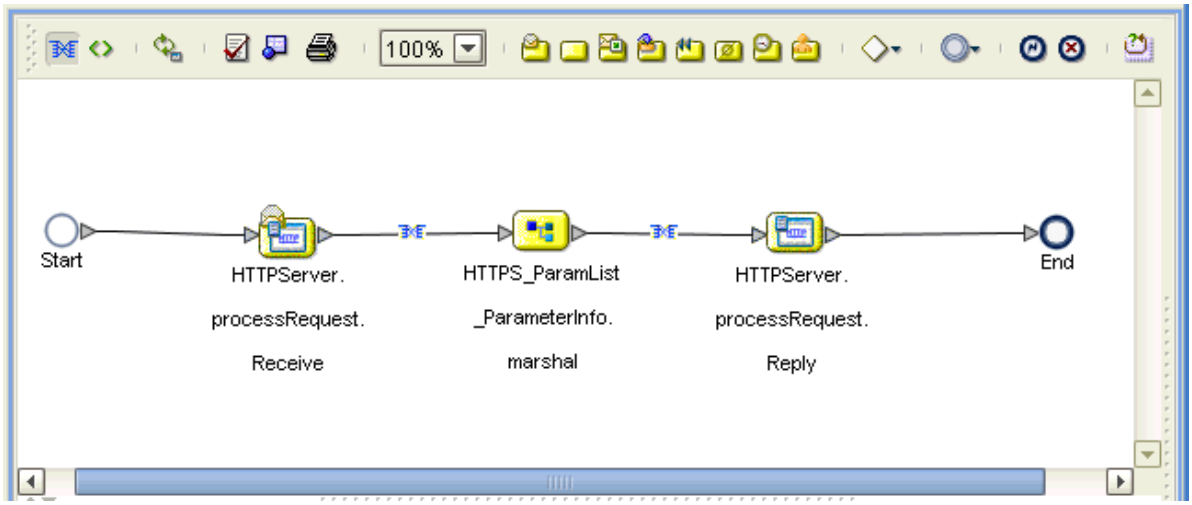
6 By dragging from one icon to another, link the icons on the canvas, as shown in Figure 30.

Figure 30 Business Process Icons With Links: Server



7 You must add two Link Business Rules (represented by a small blue, star-shaped icons) to the appropriate links, as shown in Figure 31. To do this operation, right-click on the desired link and choose **Add Business Rule** from the pop-up menu. See Figure 31 for the appropriate links where you must add the Business Rules.

Figure 31 Business Process Icons With Server Business Rules



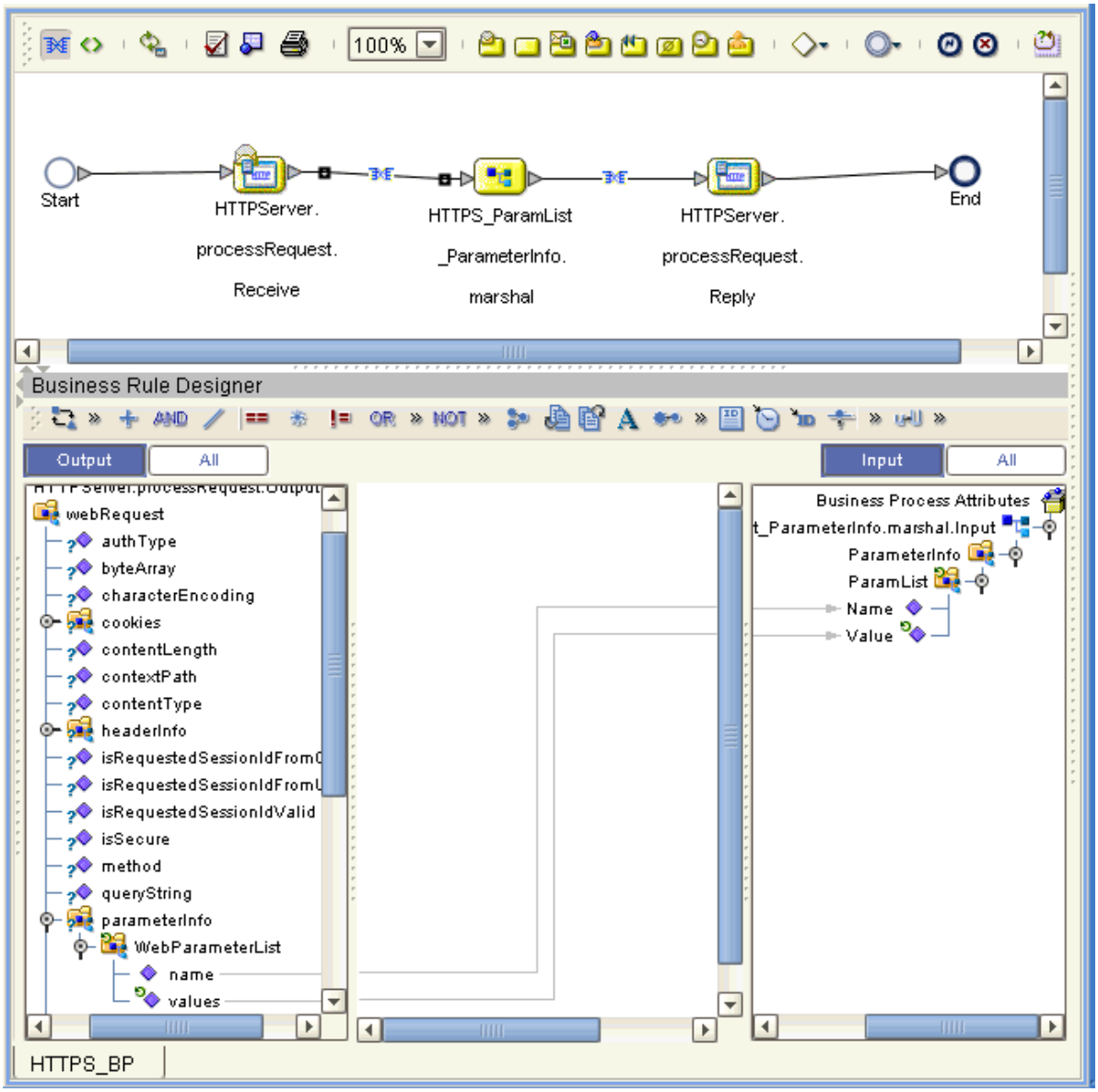
For the Business Rules, you must create the settings you want in the Business Rule Designer.

8 Select the first (left) Business Rule, for the receive operation, then click the **Map Business Process Attributes** icon in the toolbar.

The Business Rule Designer pane appears at the bottom of the window. Use the Business Rule Designer to create your Business Rules.

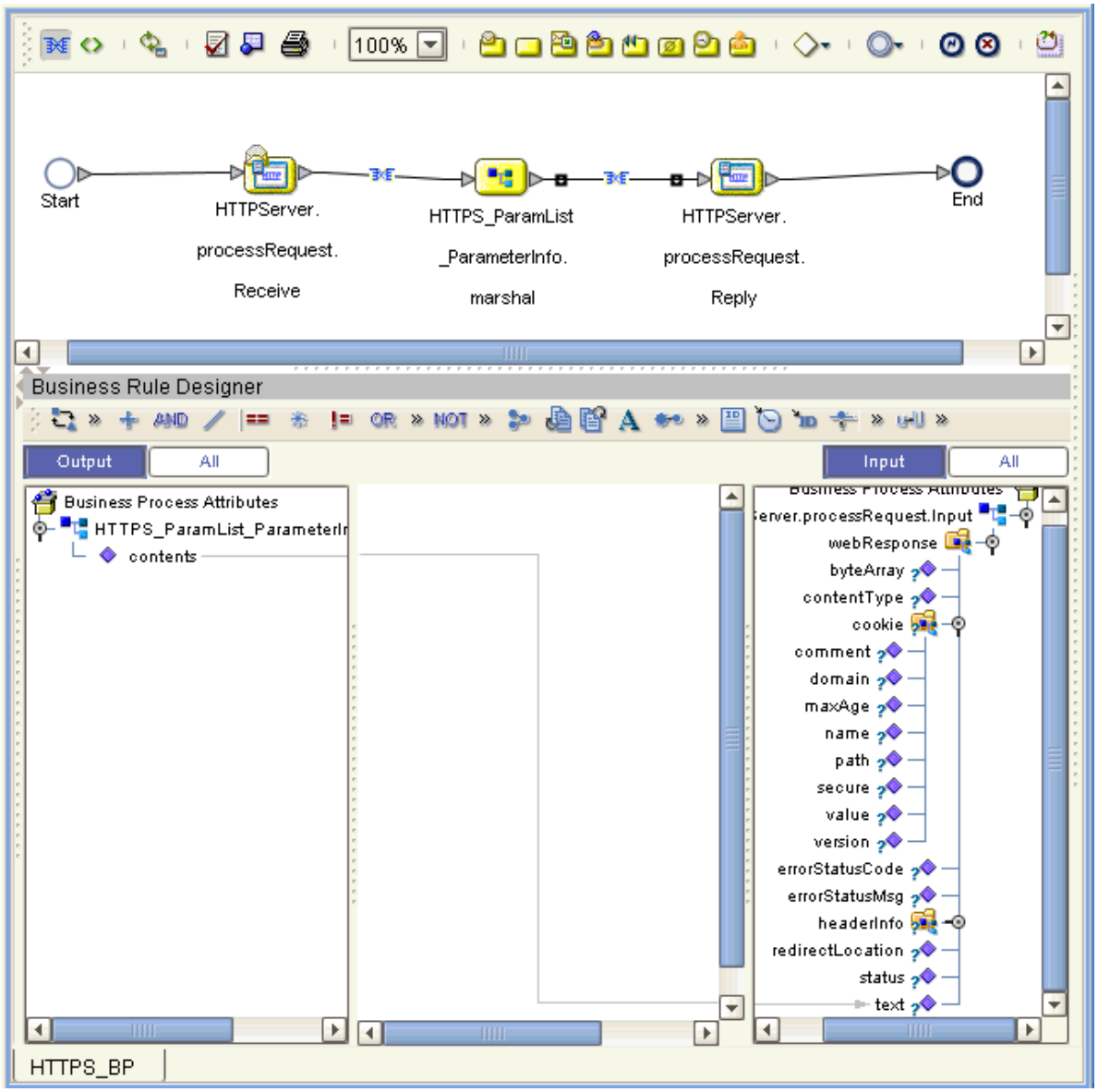
9 From the **Output** pane, drag the **name/value** pair nodes (under **WebParameterList**) to the **name/value** pair nodes (under **ParamList**) n the **Input** pane. See Figure 32.

Figure 32 Business Rule Designer: Server Receive Business Rule



10 From the **Output** pane, drag the **contents** node to the **text** node (under **headerInfo**) n the **Input** pane. See Figure 33.

Figure 33 Business Rule Designer: Server Receive Business Rule



You have now finished creating your Business Rules for the Project.

11 Click **Save** to save your Business Process.

After you have finished creating your Business Process, you can use it to define one or more of the eGate Services on your Connectivity Map.

4.5.6 Populating the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas. This operation creates the components for you.

For this sample Project, drag and drop the following components onto the Connectivity Map canvas.

- One Service, **Service1**
- HTTP(S) eWay/server external application; rename to **HTTPServer1**

Be sure to save the new Connectivity Map before you proceed. You can click **Save** for this purpose.

4.5.7 Defining the Business Process

After you have created the Connectivity Map, you need to define the Business Process Service by associating it with the appropriate Business Process in the Connectivity Map.

To do this operation, drag and drop the Business Process icon **HTTPS_BP** from the **Project Explorer** tree onto **HTTPS_BP1** in the Connectivity Map.

If the Business Process is successfully defined, the "gears" icon changes from red to yellow.

4.5.8 Binding OTDs in Business Processes

After you have defined the Business Process, you need to bind, that is, associate the appropriate OTDs with the desired components. See the *eGate Integrator User's Guide* for details.

4.5.9. Creating the Environment

See **"Creating the Project's Environment" on page 60** for details on this operation. For this Project, add the **HTTP Server** external system to the Project's Environment.

4.5.10 Setting eWay Properties

You must set the eWay properties for your specific system and for the current Project, using the eGate Enterprise Designer. For directions on accessing and using the eWay **Properties** dialog box, as well as a complete explanation of the HTTP(S) eWay properties, see **Chapter 3**.

To set properties for the File eWay

- Be sure to set the **Input file name** property to .html and the **Directory** property to the path location where you put the **postHTTPS** file.
- You can use the default or any other directory for your output **Directory**.

To set properties for the HTTP(S) eWay

1. To change the default properties for the HTTP(S) eWay, click the **HTTPServer1** external application's eWay icon on the Connectivity Map.

   The eWay **Properties** dialog box appears.

2 From the upper left pane of the eWay **Properties** dialog box, select the **HTTP Server External Configuration** folder, then the desired subfolders.

The properties settings appear in the **Properties** pane on the left.

3 Use the following setting for the **servlet-url** property:

HttpServerSample

4 Click **OK** to close the window and save.

5 Set properties on the Project **Environment**, including SSL and authentication, as desired. For instructions on how to do this operation, see **procedure on page 17**.

Project deployment

For information on how to deploy your Project, see the next section.

4.6 Deploying a Project

This section provides general procedures for Project deployment.

4.6.1 Before Activating a Project

If you have enabled the eWay's SSL feature, you must be sure that your Logical Hosts' Java Software Development Kit (SDK) versions match. See **"Verify hostname" on page 25** for details.

4.6.2 Basic Steps

For a complete explanation of how to deploy and run an eGate Project, see the *eGate Integrator User's Guide.*

To deploy the Project

1 From the **Project Explorer**, select the current Project and right-click, choosing **New > Deployment Profile** from the pop-up menus.

2 From the **Create a Deployment Profile** dialog box, enter the name of the current Project and select the Environment you created for this Project.

3 Click **OK**.

The Deployment Profile canvas appears as follows:

- The Project's external applications and Services show up as icons on the left side of the canvas.
- The external systems and Logical Host you created under **"Creating the Project's Environment" on page 60** show up as windows on the right side of the canvas.

4 Set up your Deployment Profile by dragging the icons on the left into the corresponding windows on the right.

5 Click **Save All** then **Activate**.

When the Project has been activated, a pop-up message window appears stating the activation was successful.

For additional information, see the *eGate Integrator User's Guide* and *SeeBeyond ICAN Suite Deployment Guide.*

To run the Project

For instructions on how to run a Project, see the *eGate Integrator User's Guide*.

4.6.3 Alerting and Logging

See **"Alerting and Logging" on page 62** for details.

Chapter 5

Reviewing Project With Collaborations

This chapter describes how to implement the HTTP(S) eWay using a review of the sample Project, which operates using Java-based Collaborations. This Project is included with the eWay.

This chapter assumes that you are already familiar with eGate and eInsight concepts and that you understand the basics of creating a Project using the eGate Enterprise Designer.

For a complete explanation of how to use the eGate Enterprise Designer to create and set properties for the components of an eGate Project, see the *eGate Integrator User's Guide*.

What's in This Chapter

5.1 eGate Project Description

This section provides an overview of the eGate Java Collaboration-based sample Project for the HTTP(S) eWay and how to import and use it.

5.1.1 Projects and the Enterprise Designer

A Project contains all of the eGate components you designate to perform one or more desired processes in eGate. Each eGate Project is created using the Enterprise Designer's Project canvas.

The Project canvas contains windows that represent the various stages of your Project. The types of windows in your Project canvas area include:

- **Connectivity Map Canvas**: Contains the eGate business logic components, such as Collaborations, Topics, Queues, and eWays, that you include in the structure of the Project.
- **OTD Editor**: Contains the source files used to create Object Type Definitions (OTDs) to use with a Project.
- **Collaboration Editor (Java)**: Allows you to create and/or modify Business Rules to implement the business logic of a Project's Java-enabled Collaboration Definition.

5.1.2 Importing Sample Project

Before you can view or work with a sample Project, you must first import it into eGate, using the Enterprise Designer.

Note: *The sample* ***.zip*** *file you first download may contain more than one Project and/or additional files. If this is the case, you must unzip this file first, find the desired Project file, then import the Project file. For details on how to download this file, see the* ***SeeBeyond ICAN Suite Installation Guide****.*

You are looking for the container file **HTTPS_eWay_Sample.zip**. The file name of the sample file is **HTTPS_Sample_Project_JCE.zip**.

You can name the imported Project as desired. For details on how to import a project into eGate, see the **procedure on page 39**.

5.1.3 Basic eWay Components

The HTTP(S) eWay operates based on two components, set properties and an OTD.

HTTP(S) eWay Properties

The properties for the HTTP(S) eWay contain the settings you can use to connect to a specific external system. These properties are set using the eWay **Properties** dialog box. For more information about the HTTP(S) eWay properties and the eWay **Properties** dialog box see **Chapter 3**.

HTTP OTD

The HTTP OTD maps input and output message segments at the field level. For more information on the eWay's OTD, see **Chapter 6**.

5.1.4 Sample Project Overview

The HTTP(S) eWay Java Collaboration-based sample Project demonstrates how the HTTP(S) eWay uses the GET and POST commands to request and receive data from a specific web site. The data result is received from the Web site and is sent to the following locations:

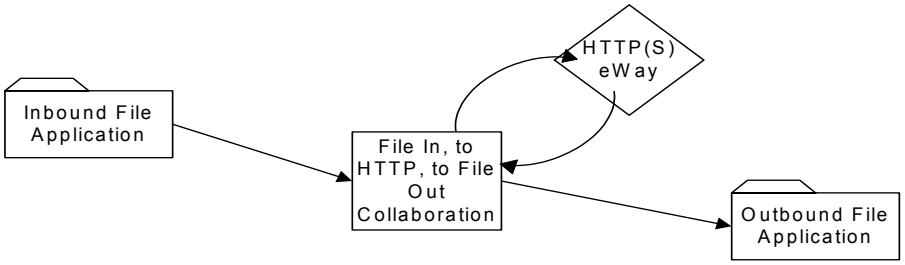
- A log file (using the **System.out.println** method) to confirm that the HTTP(S) eWay correctly requests and receives the result from the desired Web site
- An text file written to an external system via an outbound File eWay, to show the returned data and to confirm that the Project is operating correctly.

The Project has the following outputs:

- **GET Operations**: Returns the retrieved data in an .**html** file.
- **POST Operations**: Posts a name/value pair to a form and writes the same information to an .**html** file, to confirm the posting.

Figure 34 shows the flow of the sample HTTP(S) eWay Project.

Figure 34 HTTP(S) eWay Sample Project (Java Collaboration Based)



The location of input and output files are defined by the File eWay properties. By default, the inbound File eWay reads from **c:\temp\input*.txt**. The default is changed for the Project's outbound File eWay, which sends the resulting data to **c:\temp\output%d.html** (**%d** represents the serial index starting with integer 0).

The HTTP(S) eWay sample Project demonstrates how the HTTP(S) eWay processes information from an HTTP(S) system. Resulting or confirming information is then written to a text file. This scenario is illustrated in Figure 34.

Project Components

The Project has the following components:

- External file system (inbound): **FileIn**
- Inbound File eWay
- Collaboration (Java) for processing data: **HttpJCEService** (created from the **Service** icon)
- HTTP(S) eWay
- HTTP(S) external application: **HTTP_CLIENT**
- Outbound File eWay
- External file system (outbound): **FileOut**

Project Operation

The Project operates as follows:

- **FileIn**: The external file system that provides instructions to the inbound File eWay; this eWay gets a text file containing the instructions and passes them to a Collaboration (Java), **HttpJCEService**.
- **HttpJCEService**: Sends instructions to the desired HTTP(S) system via the HTTP(S) eWay. **HttpJCEService** also receives the information from the HTTP(S) system, via the HTTP(S) eWay, then sends it to a File eWay, **FileOut**.
- **HTTP_CLIENT**: The HTTP(S) external application or system; the HTTP(S) eWay handles inbound and outbound communication with this system.
- **FileOut**: The external file system that receives the information via HTTP(S); another File eWay writes the received information to a text file on this system.

Input and Output Data

The HTTP(S) eWay Project uses the following data files:

- **Get_Sample.xml**
- **Post_Sample.xml**
- **Sample_In.dtd**

These files have the following content:

GET Command: Get_Sample.xml

The input data file for the GET command is:

```
<website>
    <method>GET</method>
    <url>http://www.yahoo.com</url>
    <data/>
</website>
```

POST Command: Post_Sample.xml

The input data file for the POST command is:

```
<website>
    <method>POST</method>
    <url>http://localhost:12000/examples/servlet/
        RequestParamExample</url>
    <data>firstname^MyFirstName|lastname^MyLastName</data>
</website>
```

Sample_In DTD: Sample_In.dtd

The eGate OTD wizard is used to create a DTD-based OTD. The input data file specifies an URL for HTTP commands. The XML DTD code for this sample input data file is:

```
<!ELEMENT website (method, url, data)>
<!ELEMENT method (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT data (#PCDATA)>
```

The **Sample_In.dtd** file defines the following elements:

- **Method**: Defines whether the file is for a GET or POST command.
- **URL**: Defines the address of the target HTTP server.
- **Data**: Stores the data string(s) used in the POST command. You can use a single input string in this case.

If your input comes with a name-and-value pair (for example, user name and password fields), you can use '|' as a delimiter between pairs of data and use '^' as a sub-delimiter. For example, if the user name field is **myname**, and the password field is **mypass**, then the data element is:

```
username^myuser|password^mypass
```

You can use any number of pairs in this case. When the HTTP(S) eWay sends out the POST request, the URL becomes:

```
url?username=myuser&password=mypass
```

Where **url** is the URL element in the input file.

5.2 Summary: Sample Collaboration (Java) Project

This section explains generally how to implement the HTTP(S) eWay using the eGate Project sample with Java-based Collaborations. This sample is included on your installation CD-ROM. The sample is named **HTTP_Client_Project_JCE**. It allows you to observe an end-to-end data-exchange scenario involving eGate and the HTTP(S) eWay, in the client mode.

For instructions on how to import the sample Project, see the **procedure on page 39**. For an overview of this sample Project and what is does, see **"Sample Project Overview" on page 78**.

5.2.1 Creating Collaboration Definitions

The eGate Enterprise Designer contains a Collaboration Definition wizard (Java) that allows you to create Java-based Collaborations. You must use the wizard to create a Collaboration Definition before implementing the Collaboration.

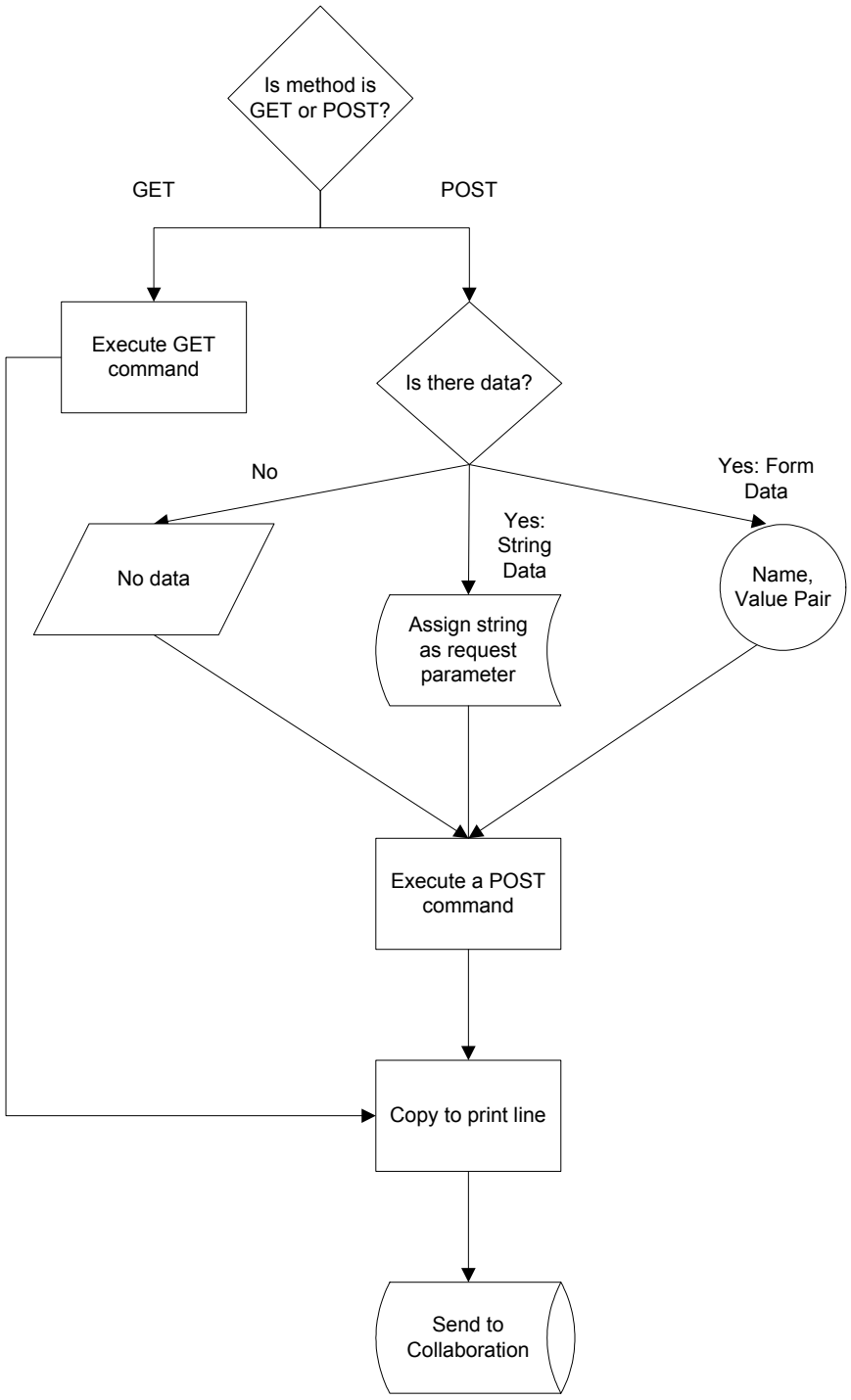
5.2.2 Using the Collaboration Editor (Java)

The Collaboration Editor (Java) window displays in the Enterprise Designer after you create a Java-based Collaboration Definition. You can also open this Editor by right-clicking on the name of the desired Collaboration Definition in the **Project Explorer** and choosing **Open** from the pop-up menu.

This Editor user interface allows you to create the Business Rules that implement your business logic for a Java-based Collaboration. You can create the desired Business Rules for your Project by dragging and dropping values from a source OTD onto the nodes of a destination HTTP(S) OTD and other OTDs. HTTP(S) OTD nodes represent HTTP(S) functions, which are in turn able to call HTTP(S) eWay methods.

The logic of the Collaboration is shown in **Figure 35 on page 82**.

Figure 35 Logic of the Collaboration (Java)



See the *eGate Integrator User's Guide* for complete information on how to use the Collaboration Editor (Java).

5.3 Creating the Project's Environment

This section provides general procedures for creating an Environment for your Project. For a complete explanation, see the *eGate Integrator User's Guide.*

To create an Environment

1. From the Enterprise Designer, click the **Environment Explorer** tab on the Enterprise Explorer.
2. Under the current **Repository** icon in the **Environment Explorer**, create a new environment for your Project and name it as desired.
3. In the **Environment Explorer**, right-click the **Environment** icon and select the desired external systems from the pop-up menu. Include external systems for the HTTP(S) eWay and the File eWays (inbound and outbound). Give them the same names as you did the corresponding external applications on the Connectivity Map.
4. Use the same pop-up menu to create a Logical Host for your Project, and name it as desired.
5. Click **Save** and return to the **Project Explorer** tab.

5.4 Setting eWay Properties

You must set the eWay properties for your specific system and for the current Project, using the eGate Enterprise Designer. For directions on accessing and using the eWay **Properties** dialog box, as well as a complete explanation of the HTTP(S) eWay properties, see **Chapter 3**.

To set properties for the File eWays

1. From the **Project Explorer**, open the Connectivity Map for the sample Project.
2. To change the default properties for the inbound File eWay, double-click the **FileIn** external system's eWay icon. For this eWay, select the **Inbound** properties.

   The eWay **Properties** dialog box appears. This eWay can use the current default properties settings for the sample Project. However, if you are using different file names and/or folders than the defaults, you must enter the names of the files/folders you are using.

Note: *Even if you do not change the eWay's properties, you must open each* ***Properties*** *dialog box for every eWay and click* ***OK*** *to activate the eWay.*

3. For this sample, use the **C:\temp** as the property for **Directory**, and for **Input file name**, enter ***.txt**.
4. Click **OK** to save the settings and close the eWay **Properties** dialog box.
5. To change the default properties for the outbound File eWay, double-click the **FileOut** external system's eWay icon. For this eWay, select the **Outbound** properties.

Use your file and folder names if they differ from the defaults. For all other properties, you can use the defaults.

6 Click **OK** to close the dialog box and save.

To set properties for the HTTP(S) eWays

1 To begin changing the default properties for the HTTP(S) eWay, double-click the **HTTP_CLIENT** external system's eWay icon.

The eWay **Properties** dialog box appears.

2 The properties settings appear in the **Properties** pane on the left.

3 Set the properties as desired then click **OK** to close the dialog box and save.

4 From the Enterprise Designer, click the **Environment Explorer** tab.

5 In the **Environment Explorer**, right-click the new Environment you created for your Project, and select **New HTTP External System**. Give this system the same name as you gave the corresponding external application (**HTTP_CLIENT**) created on the Connectivity Map in the **Project Explorer**.

For details on how to do these operations, see the *eGate Integrator User's Guide*.

6 In the left pane, right-click the **HTTP_CLIENT** external system icon and select **Properties** from the pop-up menu.

The eWay **Properties** dialog box appears. Use this dialog box to set additional properties settings.

7 The properties settings appear in the **Properties** pane on the left.

8 For the settings, enter the information appropriate to your system.

9 Click **OK** to close the dialog box and save.

5.5 Deploying a Project

for instructions on how to deploy a Project, see **"Deploying a Project" on page 74**.

Chapter 6

Understanding the HTTP(S) eWay OTD

This chapter provides an overview of OTDs and describes the HTTP(S) eWay's Object Type Definition (OTD) structure.

What's in This Chapter

6.1 Overview of eWay OTDs

An OTD contains a set of rules that define an object. The object encodes data as it travels through eGate. OTDs are used as the basis for creating a Java-based Collaboration Definition for a Project.

Each OTD acts as a template with a unique set of eWay features. The HTTP(S) eWay OTD template is not customizable and cannot be edited.

The basic parts of an OTD are:

- **Element**: This is the highest level in the OTD tree. The element is the basic container that holds the other parts of the OTD. The element can contain fields and methods.
- **Field**: Fields are used to represent data. A field can contain data in any of the following formats: **string**, **boolean**, **int**, **double**, or **float**.
- **Method**: Method nodes represent actual Java methods.
- **Parameters**: Parameters nodes represent the Java methods' parameters.

6.1.1 OTD Components

Each OTD is made up of the following components:

- **OTD Operation**: The OTD is used in a Collaboration Definition to operate with eWays.
- **Definition and eGate Enterprise Designer**: An eWay **Properties** dialog box provides a central location where you can define the eWay's properties. The Enterprise Designer also allows you to access this window.
- **eWay**: An eWay provides access to the information necessary to interface with a specified external application.

All OTDs must be set up and administered using the Enterprise Designer.

Note: *For complete information on how to use the Enterprise Designer and the eWay eWay* ***Properties*** *dialog box, see the* ***eGate Integrator User's Guide****.*

Client components

Any client components relevant to these OTDs have their own requirements. See the client system's documentation for details.

6.2 HTTP(S) OTD

The HTTP(S) OTD is specific to the HTTP(S) eWay. It is used as an inbound or outbound OTD in a Collaboration.

OTDs have a tree-like hierarchical data structure composed of fields containing methods and properties.

The top root element of the OTD is the **HTTPClientApplication** interface, and the fields underneath contain Java methods. You can use these Java methods to create Business Rules that specify the HTTP message format and invoke messaging to and/or from an HTTP server.

To access other Java classes and methods, you can use the Collaboration Editor (Java) to utilize the entire contents available for **HTTPClientApplication**.

6.2.1 HTTP OTD Method Descriptions

The HTTP OTD includes the following methods used in HTTP data exchange:

get
The method called in the Collaboration (Java) to send an HTTP **get** request to an HTTP server.

post
The method called in the Collaboration (Java) to send an HTTP **post** request to an HTTP server.

getRequest
The method called in the Collaboration (Java) for other "request" related helper methods, such as to set the URL, to add properties, etc.

getResult
The method called in the Collaboration (Java) for other "respond" related helper methods, such as, to obtain the respond code, respond result, text result, and so on.

For more information on methods available in the HTTP OTDs, see the HTTP(S) eWay's **Javadoc**.

Chapter 7

Operating SSL

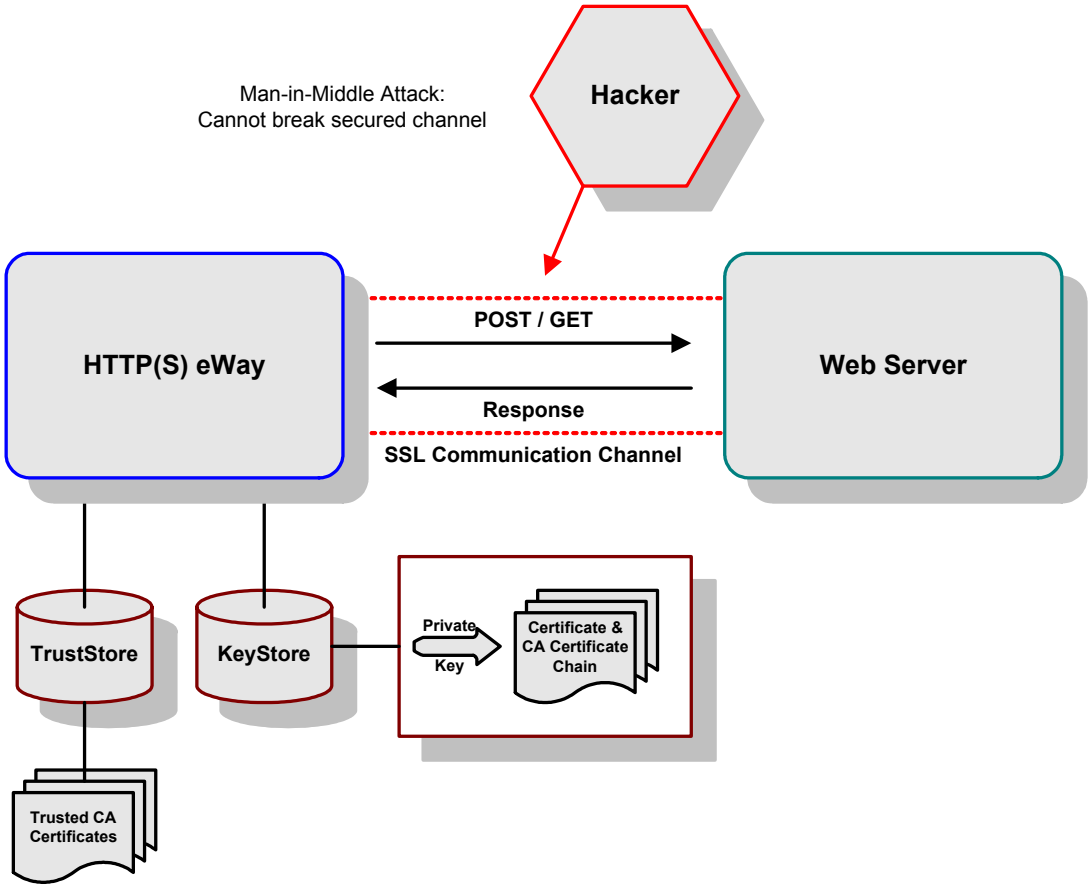
This chapter explains the operation of the Secure Sockets Layer (SSL) feature available with the HTTP(S) eWay.

What's in This Chapter

7.1 Overview

The use of SSL with HTTP, here called HTTP(S), enables HTTP data exchanges that are secure from unauthorized interception from "hackers" or other entities. The eWay's SSL feature provides a secure communications channel for the data exchanges (see **Figure 36 on page 88**).

Figure 36 General SSL Operation: HTTPS



This SSL feature is supported through the use of JSSE version 1.0.3.

Note: *JSSE 1.0.3 if using SDK 1.3.1, or JSEE version bundled with SDK 1.4.1 release.*

Currently, the JSSE reference implementation is used. JSSE is a provider-based architecture, meaning that there is a set of standard interfaces for cryptographic algorithms, hashing algorithms, secured-socket-layered URL stream handlers, and so on.

Because the user is interacting with JSSE through these interfaces, the different components can be mixed and matched as long as the implementation is programmed under the published interfaces. However, some implementations may not support a particular algorithm.

The JSSE 1.0.3 application programming interface (API) is capable of supporting SSL versions 2.0 and 3.0 and Transport Layer Security (TLS) version 1.0. These security protocols encapsulate a normal bidirectional stream socket and the JSSE 1.0.3 API adds transparent support for authentication, encryption, and integrity protection. The JSSE reference implementation implements SSL version 3.0 and TLS 1.0.

For more information, visit the Sun Java Web site at the following URL:

http://java.sun.com

Note: *See the JSSE documentation provided by Sun Microsystems for further details.*

7.2 KeyStores and TrustStores

As depicted in Figure 36, JSSE makes use of files called *KeyStores* and *TrustStores*. The KeyStore is used by the eWay for client authentication, while the TrustStore is used to authenticate a server in SSL authentication.

- A *KeyStore* consists of a database containing a private key and an associated certificate, or an associated certificate chain. The certificate chain consists of the client certificate and one or more certification authority (CA) certificates.
- A *TrustStore* contains only the certificates trusted by the client (a "trust" store). These certificates are CA root certificates, that is, self-signed certificates. The installation of the HTTP(S) eWay installs a TrustStore file named **trustedcacertsjks**, which can be used as the TrustStore for the eWay.

Both KeyStores and TrustStores are managed by means of a utility called **keytool**, which is a part of the Java SDK installation.

Java SDK version special considerations

To use **keytool** with the Java SDK version 1.3, you must include **jcert.jar, jnet.jar**, and **jsse.jar** in your CLASSPATH. With the Java SDK version 1.4, you do not have to install these .**jar** files because they are part of this version.

With the Java SDK version 1.3, you must add the following line to the file **jre\lib\security\java.security**:

```
security.provider.3=com.sun.net.ssl.internal.ssl.Provider
```

However, with the Java SDK version 1.4, you do not have to edit this file.

See the installation manual for the JSSE version 1.0.3 for more information.

7.2.1 Generating a KeyStore and TrustStore

This section explains steps on how to create both a KeyStore and a TrustStore (or import a certificate into an existing TrustStore such as **trustedcacertsjks**). The primary tool used is **keytool**, but **openssl** is also used as a reference for generating **pkcs12** KeyStores.

For more information on **openssl**, and available downloads, visit the following Web site:

http://www.openssl.org.

7.2.2 KeyStores

This section explains how to use KeyStores.

Creating a KeyStore in JKS Format

This section explains how to create a KeyStore using the JKS format as the database format for both the private key, and the associated certificate or certificate chain. By default, as specified in the java.security file, **keytool** uses JKS as the format of the key and certificate databases (KeyStore and TrustStores). A CA must sign the certificate signing request (CSR). The CA is therefore trusted by the server-side application to which the eWay is connected.

To generate a KeyStore

Use the following command:

```
keytool -keystore clientkeystore -genkey -alias client
```

You are prompted for several pieces of information required to generate a CSR. A sample key generation section follows:

```
Enter keystore password: seebeyond
What is your first and last name?
[Unknown]: development.seebeyond.com
What is the name of your organizational unit?
[Unknown]: Development
what is the name of your organization?
[Unknown]: SeeBeyond
What is the name of your City of Locality?
[Unknown]: Monrovia
What is the name of your State or Province?
[Unknown]: California
What is the two-letter country code for this unit?
[Unknown]: US
Is<CN=Foo Bar, OU=Development, O=SeeBeyond, L=Monrovia,
ST=California, C=US> correct?
[no]: yes

Enter key password for <client>
    (RETURN if same as keystore password):
```

If the KeyStore password is specified, then the password must be provided for the eWay. Press RETURN when prompted for the key password (this action makes the key password the same as the KeyStore password).

This operation creates a KeyStore file **clientkeystore** in the current working directory. You must specify a fully-qualified domain for the "first and last name" question. The reason for this use is that some CAs such as Verisign expect this properties to be a fully qualified domain name.

There are CAs that do not require the fully qualified domain, but it is recommended to use the fully-qualified domain name for the sake of portability. All the other information given must be valid. If the information can not be validated, a CA such as Verisign does not sign a generated CSR for this entry.

This KeyStore contains an entry with an alias of **client**. This entry consists of the Generated private key and information needed for generating a CSR as follows:

```
keytool -keystore clientkeystore -certreq alias client -keyalg rsa
    -file client.csr
```

This command generates a certificate signing request which can be provided to a CA for a certificate request. The file **client.csr** contains the CSR in PEM format.

Some CA (one trusted by the Web server to which the eWay is connecting) must sign the CSR. The CA generates a certificate for the corresponding CSR and signs the certificate with its private key. For more information, visit the following Web sites:

http://www.thawte.com

or

http://www.verisign.com

If the certificate is chained with the CA's certificate, perform step 1; otherwise, perform step 2 in the following list:

1 The following command assumes the client certificate is in the file **client.cer** and the CA's certificate is in the file **CARoot.cer**:

```
keytool -import -keystore clientstore -file client.cer -alias
    client
```

This command imports the certificate (which can include more than one CA in addition to the Client's certificate).

Also use the following command to import the CA's certificate into the KeyStore for chaining with the client's certificate:

```
keytool -import -keystore clientkeystore -file CARootcer -alias
    theCARoot
```

2 The following command imports the client's certificate signed by the CA whose certificate was imported in the preceding step:

```
keytool -import -keystore clientkeystore -file client.cer -alias
    client
```

The generated file **clientkeystore** contains the client's private key and the associated certificate chain used for client authentication and signing. The KeyStore and/or **clientkeystore**, can then be used as the eWay's KeyStore.

See the **"KeyStores" on page 89** for more information.

Creating a KeyStore in PKCS12 Format

This section explains how to create a PKCS12 KeyStore to work with JSSE. In a real working environment, a customer could already have an existing private key and certificate (signed by a known CA). In this case, JKS format can not be used, because it does not allow the user to import/export the private key through **keytool**. It is necessary to generate a PKCS12 database consisting of the private key and its certificate.

The generated PKCS12 database can then be used as the eWay's KeyStore. The **keytool** utility is currently lacking the ability to write to a PKCS12 database. However, it can read from a PKCS12 database.

Note: *There are additional third-party tools available for generating PKCS12 certificates, if you want to use a different tool.*

For the following example, **openssl** is used to generate the PKCS12 KeyStore:

```
cat.mykey.pem.txt mycertificate.pem.txt>mykeycertificate.pem.txt
```

The existing key is in the file **mykey.pem.txt** in PEM format. The certificate is in **mycertificate.pem.txt**, which is also in PEM format. A text file must be created which contains the key followed by the certificate as follows:

```
openssl pkcs12 -export -in mykeycertificate.pem.txt -out
    mykeystore.pkcs12 -name myAlias -noiter -nomaciter
```

This command prompts the user for a password. The password is required. The KeyStore fails to work with JSSE without a password. This password must also be supplied as the password for the eWay's KeyStore password (see **"KeyStore password" on page 24**).

This command also uses the **openssl pkcs12** command to generate a PKCS12 KeyStore with the private key and certificate. The generated KeyStore is **mykeystore.pkcs12** with an entry specified by the **myAlias** alias. This entry contains the private key and the certificate provided by the **-in** argument. The **noiter** and **nomaciter** options must be specified to allow the generated KeyStore to be recognized properly by JSSE.

7.2.3 TrustStores

Creating a TrustStore

For demonstration purposes, suppose you have the following CAs that you trust: **firstCA.cert, secondCA.cert, thirdCA.cert**, located in the directory **C:\cascerts**. You can create a new TrustStore consisting of these three trusted certificates.

To create a new TrustStore

Use the following command:

```
keytool -import -file C:\cascerts\firstCA.cert -alias firstCA
    -keystore myTrustStore
```

You must enter this command two more times, but for the second and third entries, substitute **secondCA** and **thirdCA** for **firstCA**. Each of these command entries has the following purposes:

1. The first entry creates a KeyStore file name **myTrustStore** in the current working directory and imports the **firstCA** certificate into the TrustStore with an alias of **firstCA**. The format of **myTrustStore** is JKS.
2. For the second entry, substitute **secondCA** to import the **secondCA** certificate into the TrustStore, **myTrustStore**.
3. For the third entry, substitute **thirdCA** to import the **thirdCA** certificate into the TrustStore.

Once completed, myTrustStore is available to be used as the TrustStore for the eWay. See **"TrustStores" on page 92** for more information.

Using an Existing TrustStore

This section explains how to use an existing TrustStore such as trustedcacertsjks. Notice that in the previous section, steps 2 and 3 were used to import two CAs into the TrustStore created in step 1.

For example, suppose you have a trusted certificate file named: **C:\trustedcerts\foo.cert** and want to import it to the **trustedcacertsjks** TrustStore.

If you are importing certificates into an existing TrustStore, use:

```
keytool -import -file C:\cacerts\secondCA.cert -alias secondCA
    -keystore trustedcacertsjks
```

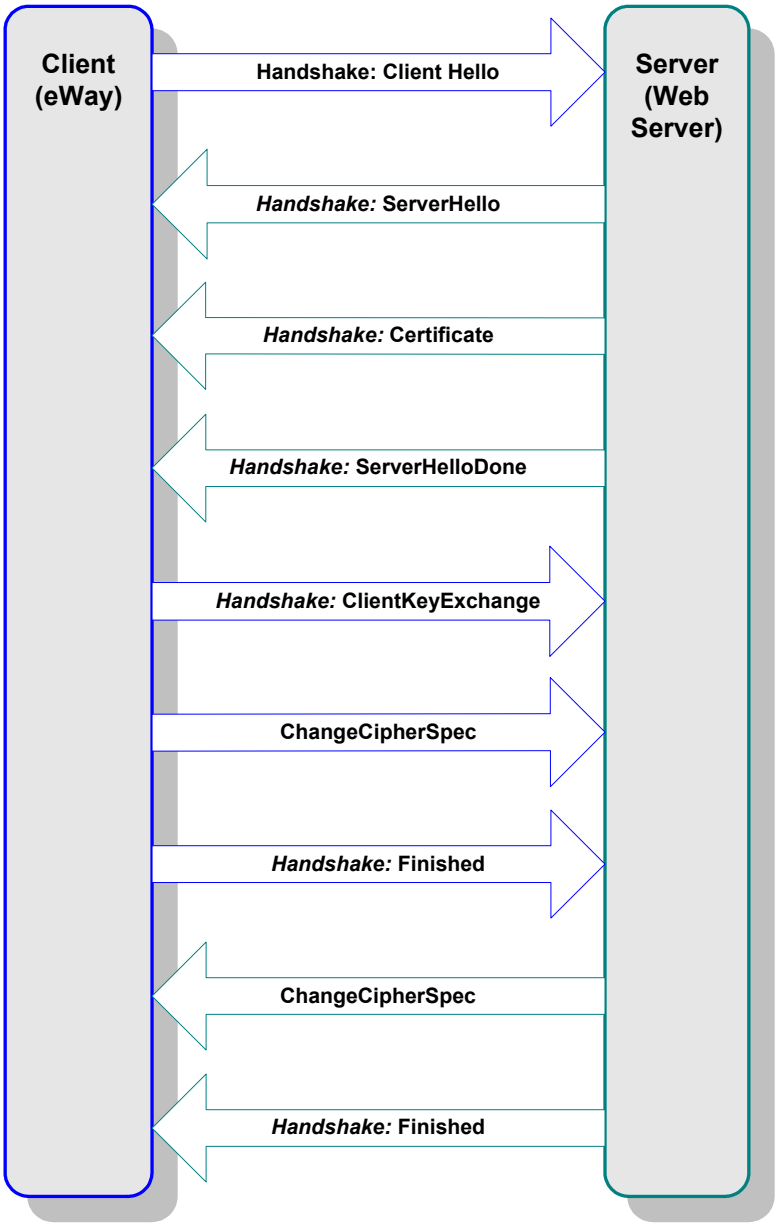
Once you are finished, **trustedcacertsjks** can be used as the TrustStore for the eWay. See **"TrustStores" on page 92** for more information.

7.3 SSL Handshaking

There are two options available for setting up SSL connectivity with a Web server:

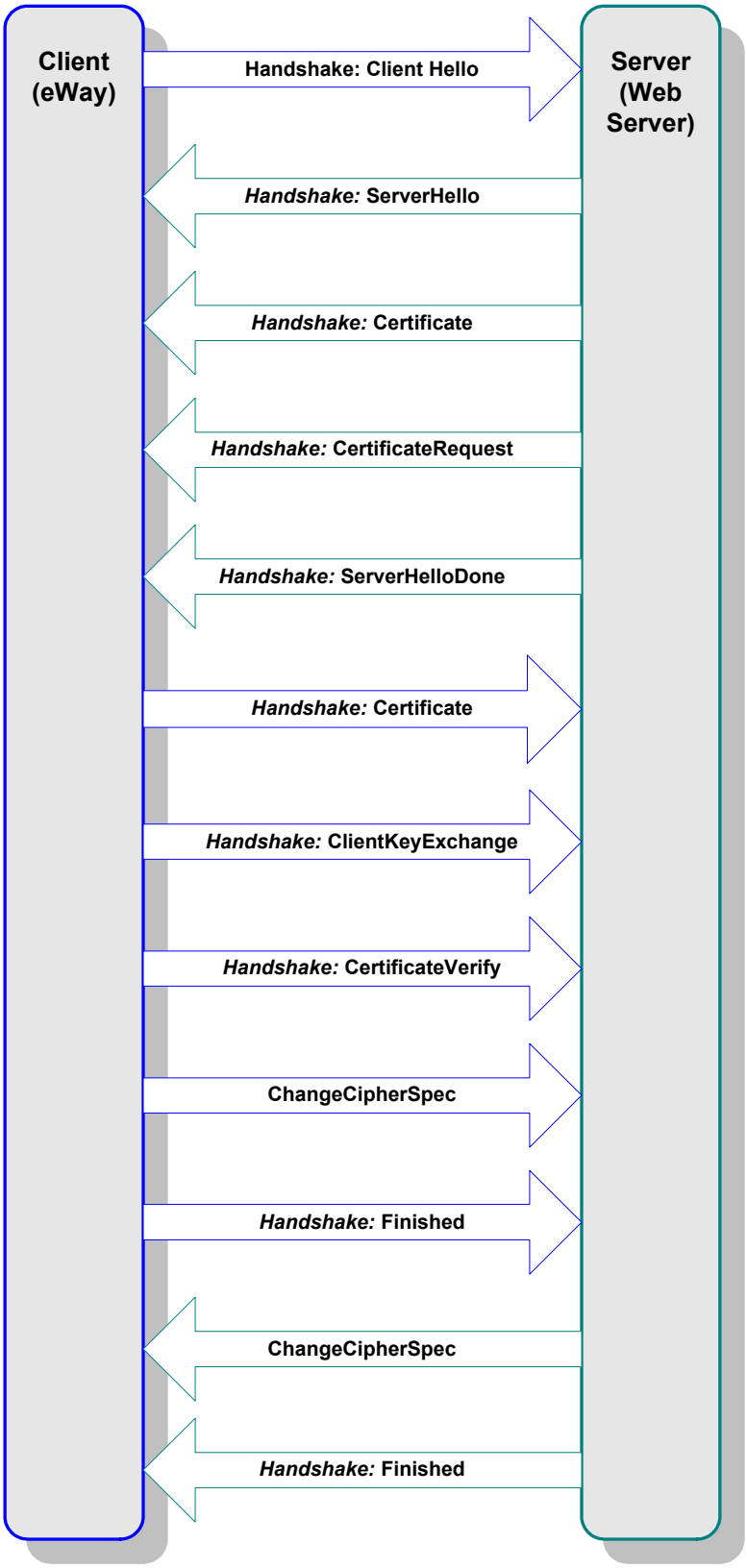
- **Server-side Authentication**: The majority of eCommerce Web sites on the Internet are configured for server-side authentication. The eWay requests a certificate from the Web server and authenticates the Web server by verifying that the certificate can be trusted. Essentially, the eWay performs this operation by looking into its TrustStore for a CA certificate with a public key that can validate the signature on the certificate received from the Web server. This option is illustrated in **Figure 37 on page 94**.

Figure 37 Server-side Authentication



- **Dual authentication**: This option requires authentication from both the eWay and Web server. The server side (Web server) of the authentication process is the same as that described previously. In addition, however, the Web server requests a certificate from the eWay. The eWay then sends its certificate to the Web server. The server, in turn, authenticates the eWay by looking into its TrustStore for a matching trusted CA certificate. The communication channel is established by the process of both parties' requesting certificate information. This option is illustrated in Figure 38 on page 95.

Figure 38 Dual Authentication



Chapter 8

Using the OpenSSL Utility

This chapter provides detailed information on how to use the **OpenSSL** utility.

What's in This Chapter

0.1 Using OpenSSL: Introduction

The **OpenSSL** utility is a free implementation of cryptographic, hashing, and public key algorithms such as 3DES, SHA1, and RSA respectively. This utility has many options including certificate signing, which **keytool** does not provide. You can download **OpenSSL** from the following Web site:

http://www.openssl.org

Follow the build and installation instruction for **OpenSSL**.

To learn more about SSL, and the high level aspects of cryptography, a good source of reference is a book entitled *SSL and TLS: Designing and Building Secure Systems* (by Eric Rescorla, Published by Addison Wesley Professional; ISBN: 0201615983).

0.2 Creating a Sample CA Certificate

The sample given in this section demonstrates the use of the **OpenSSL** utility to create a CA. This generated CA is then used to sign a CSR (see **"Signing Certificates With Your Own CA" on page 97**), whether it is generated from **keytool** or **OpenSSL**.

For testing purposes a sample CA can be generated. To avoid spending additional funds to have a commercial CA sign test certificates, a sample is generated and used to sign the test certificate.

Perform the following operations from the command line:

```
openssl  req  -config c:\openssl\bin\openssl.cnf  -new  -x509  -
keyout  ca-key.pem.txt -out  ca-certificate.pem.txt  -days  365

Using properties from c:\openssl\bin\openssl.cnf
Loading 'screen' into random state: done
Generating a 1024 bit RSA private key
..................++++++
.....................++++++
writing new private key to 'ca-key.pem.txt'
Enter PEM pass phrase:
Verifying password: Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be
    incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name
    or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []:US
State or Province Name (full name) []:California
Locality Name (eg, city) []:Monrovia
Organization Name (eg, company) []:SeeBeyond
Organizational Unit Name (eg, section) []:Development
Common Name (eg, your websites domain name) []
    :development.seebeyond.com
Email Address []:development@seebeyond.com
```

You are prompted for information. You must enter a password and remember this password for signing certificates with the CA's private key. This command creates a private key and the corresponding certificate for the CA. The certificate is valid for 365 days starting from the date and time it was created.

The properties file **C:\openssl\bin\openssl.cnf** is needed for the **req** command. The default **config.cnf** file is in the **OpenSSL** package under the **apps** sub-directory.

Note: *That to use this file in Windows, you must change the paths to use double back-slashes. See* ***"Windows OpenSSL.cnf File Example" on page 99*** *for a complete* ***Config.cnf*** *file example, which is known to work in a Windows environment.*

0.3 Signing Certificates With Your Own CA

The example in this section shows how to create a CSR with **keytool** and generate a signed certificate for the CSR with the CA created in the previous section. The steps shown in this section, for generating a **KeyStore** and a CSR, were already explained under **"Creating a KeyStore in JKS Format" on page 90**.

Note: *No details are given here for the* ***keytool*** *commands. See* ***"Creating a KeyStore in JKS Format" on page 90*** *for more information.*

To create a CSR with keytool and generate a signed certificate for the CSR

1

```
keytool -keystore clientkeystore -genkey -alias client

Enter keystore password:  seebeyond
What is your first and last name?
[Unknown]:  development.seebeyond.com
What is the name of your organizational unit?
[Unknown]:  Development
What is the name of your organization?
[Unknown]:  SeeBeyond
What is the name of your City or Locality?
[Unknown]:  Monrovia
What is the name of your State or Province?
[Unknown]:  California
What is the two-letter country code for this unit?
[Unknown]:  US
Is <CN=Foo Bar, OU=Development, O=SeeBeyond, L=Monrovia, ST=Californi
a, C=US>   correct?
[no]:  yes

Enter key password for <client>
(RETURN if same as keystore password):
```

2

```
keytool  -keystore clientkeystore  -certreq  -alias client  -
keyalg rsa  -file client.csr
```

3

```
openssl  x509  -req  -CA
    ca-certificate.pem.txt  CAkey ca-key.pem.txt
    -in client.csr -out client.cer  -days 365  -CAcreateserial
```

This is how we create a signed certificate for the associated CSR. The option **-CAcreateserial** is needed if this is the first time the command is issued. It is used to create an initial serial number file used for tracking certificate signing. This certificate will be valid for 365 days.

4

```
keytool -import -keystore clientkeystore -file client.cer
    -alias client

Enter keystore password: seebeyond
keytool error: java.lang.Exception: Failed to establish chain from
    reply
```

You get an exception because there is no certificate chain in the client certificate so we have to import the CA's certificate into the **KeyStore** first. You can then import the client.cer itself to form a certificate chain. You need the following steps:

1

```
keytool  -import  -keystore clientkeystore  -file CA
    ca-certificate.pem.txt  -alias theCARoot

Enter keystore password:  seebeyond
Owner: EmailAddress=development@seebeyond.com, CN=development.seebeyo
nd.com, OU=Development, O=SeeBeyond, L=Monrovia, ST=California, C=US
```

```
Issuer: EmailAddress=development@seebeyond.com, CN=development.seebey
ond.com,
OU=Development, O=SeeBeyond, L=Monrovia, ST=California, C=US
Serial number: 0
Valid from: Tue May 08 15:09:07 PDT 2001 until: Wed May 08
    15:09:07 PDT 2002
Certificate fingerprints:
MD5:  60:73:83:A0:7C:33:28:C3:D3:A4:35:A2:1E:34:87:F0
SHA1: C6:D0:C7:93:8E:A4:08:F8:38:BB:D4:11:03:C9:E6:CB:9C:D0:72:D0
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

2

```
keytool -import -keystore clientkeystore -file  client.cer -alias
    client

Enter keystore password: seebeyond
Certificate reply was installed in keystore
```

Now that we have a private key and an associating certificate chain in the **KeyStore clientkeystore**, we can use it as a **KeyStore** for client (eWay) authentication. The only warning is that the CA certificate must be imported into the trusted certificate store of the Web server to which you will be connecting. Moreover, the Web server must be configured for client authentication (**httpd.conf** for Apache, for example).

This appendix contains the contents of the **openssl.cnf file** that can be used on Windows. Be sure to make the appropriate changes to the directories.

0.4 Windows OpenSSL.cnf File Example

This section contains the contents of the **openssl.cnf file** that can be used on Windows. Be sure to make the appropriate changes to the directories.

```
#
# SSLeay example properties file.
# This is mostly being used for generation of certificate requests.
#

RANDFILE = .rnd

####################################################################
[ ca ]
default_ca= CA_default# The default ca section

####################################################################
[ CA_default ]

dir      = G:\\openssl\\\bin\\demoCA# Where everything is kept
certs    = $dir\\certs    # Where the issued certs are kept
crl_dir= $dir\\crl   # Where the issued crl are kept
database= $dir\\index.txt# database index file.
new_certs_dir= $dir\\newcerts# default place for new certs.
```

```
certificate= $dir\\cacert.pem    # The CA certificate
serial   = $dir\\serial    # The current serial number
crl      = $dir\\crl.pem    # The current CRL
private_key= $dir\\private\\cakey.pem   # The private key
RANDFILE= $dir\\private\\private.rnd # private random number file

x509_extensions= x509v3_extensions# The extentions to add to the cert
default_days= 365    # how long to certify for
default_crl_days= 30# how long before next CRL
default_md= md5      # which md to use.
preserve = no        # keep passed DN ordering

# A few difference way of specifying how similar the request should l
ook
# For type CA, the listed attributes must be the same, and the option
al
# and supplied fields are just that :-)
policy   = policy_match

# For the CA policy
[ policy_match ]
countryName  = match
stateOrProvinceName= match
organizationName= match
organizationalUnitName= optional
commonName   = supplied
emailAddress = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName= optional
stateOrProvinceName= optional
localityName= optional
organizationName= optional
organizationalUnitName= optional
commonName   = supplied
emailAddress = optional

####################################################################
[ req ]
default_bits= 1024
default_keyfile = privkey.pem
distinguished_name= req_distinguished_name
attributes= req_attributes

[ req_distinguished_name ]
countryName  = Country Name (2 letter code)
countryName_min= 2
countryName_max= 2

stateOrProvinceName= State or Province Name (full name)

localityName = Locality Name (eg, city)

0.organizationName= Organization Name (eg, company)

organizationalUnitName= Organizational Unit Name (eg, section)

commonName   = Common Name (eg, your website's domain name)
commonName_max= 64

emailAddress = Email Address
```

```
emailAddress_max= 40

[ req_attributes ]
challengePassword= A challenge password
challengePassword_min= 4
challengePassword_max= 20

[ x509v3_extensions ]
```

Note: *The following copyright notices apply:*

Copyright © 1998-2001 The OpenSSL Project. All rights reserved.

Copyright © 1994-2002 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. ***http://www.w3.org/Consortium/Legal/***

Index

R

S

T

U

V

W