SeeBeyond™ eBusiness Integration Suite

e*Way Intelligent Adapter for the Microsoft Internet Information Server User's Guide

Release 4.5.3

Java Version

Version 20020424101520.

Contents

Chapter 4

Chapter 5

Chapter 1

Introduction

This chapter provides an overview of SeeBeyond™ Technology Corporation's (SeeBeyond™) e*Way™ Intelligent Adapter for the Microsoft Internet Information Server (MS IIS e*Way).

1.1 Overview

The MS IIS e*Way is a gateway to the e*Gate Integrator system for the Microsoft Internet Information Server (IIS) Web server. The e*Way primarily uses the following components:

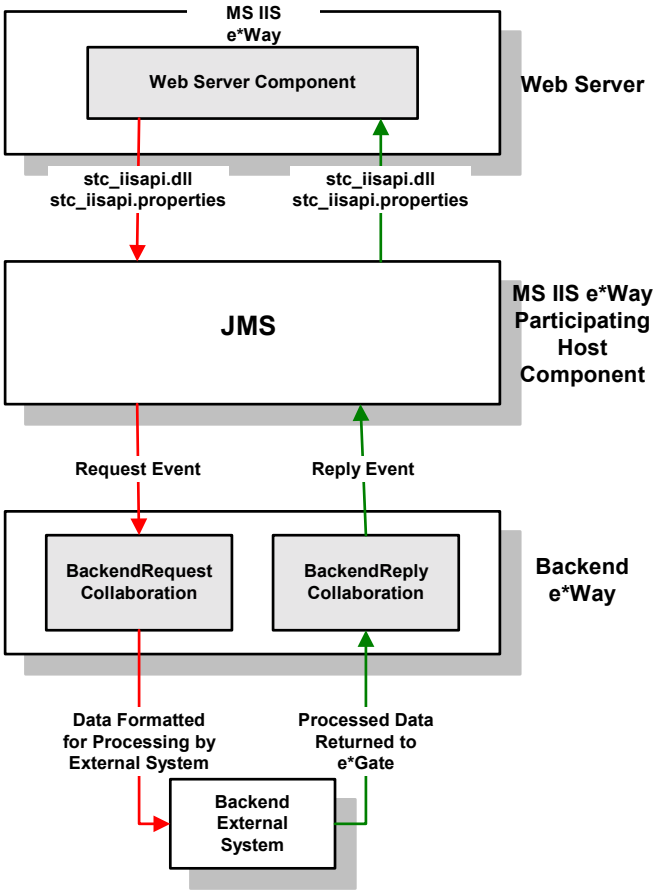
- A **.dll** file on the Web server
- SeeBeyond Java Messaging Service (JMS) on an e*Gate Participating Host

The **.dll** file's operation parses input supplied by the Web server, using either the GET or POST method, then packages the input along with the server's variables. After this process is done, the server sends the packaged message to the JMS component, which resides on the Participating Host.

After sending the message, the Web server (via the **.dll** file) waits for a reply. Upon receipt of the reply, a response message is sent to the Web server, which can deliver the message to a requesting Web client (for example, a browser).

Figure 1 on page 8 shows a sample implementation of the MS IIS e*Way. For details on implementing the MS IIS e*Way, see **Chapter 4**.

Figure 1 Overview of MS IIS Web Server e*Way Implementation



1.2 e*Way Components

The MS IIS e*Way is made up of the following components:

- Web server component
- Participating Host components

A complete list of installed files appears in **Table 1 on page 13**.

1.3 Intended Reader

The reader of this guide is presumed to be a developer or system administrator with responsibility for maintaining the e*Gate system; to have working knowledge of operations and administration for the operating systems under which the Web server and e*Gate systems run; to be familiar with MS IIS concepts; and to be familiar with Windows-style GUI operations.

1.4 Supporting Documents

The following documents are designed to work in conjunction with the *MS IIS e*Way Intelligent Adapter User's Guide* and to provide additional information that may prove useful to you.

- *Creating an End-to-end Scenario with e*Gate Integrator*
- *e*Gate Integrator Alert Agent User's Guide*
- *e*Gate Integrator Alert and Log File Reference Guide*
- *e*Gate Integrator Collaboration Services Reference Guide*
- *e*Gate Integrator Installation Guide*
- *e*Gate Integrator Intelligent Queue Services Reference Guide*
- *e*Gate Integrator SNMP Agent User's Guide*
- *e*Gate Integrator System Administration and Operations Guide*
- *e*Gate Integrator User's Guide*
- *Standard e*Way Intelligent Adapters User's Guide*
- *Readme.txt* file on the e*Gate installation CD-ROM.

1.5 Supported Operating Systems

The MS IIS e*Way is available on the following operating systems:

- Windows 2000, Windows 2000 SP1, and Windows 2000 SP2
- Windows NT 4.0 SP6a

1.6 System Requirements

To use the MS IIS e*Way, you need to meet the following requirements:

- An e*Gate Participating Host, version 4.5.1 or later.
- In addition to the disk space required by e*Gate, additional disk space is required to process and queue the data that this e*Way processes; the amount necessary can vary based on the type and size of the data being processed, as well as any external applications performing the processing.
- A TCP/IP network connection

1.7 External System Requirements

To use the MS IIS e*Way, you need to meet the following external system requirements:

- Microsoft IIS Web server, version 5.0
- Sufficient memory and disk space to support Web-server functions. See your IIS Web server user's guides for more information about server requirements.

Note: *The e*Gate Participating Host can optionally host the Web server but is not required to do so.*

Chapter 2

Installation

This chapter covers the requirements for installing the e*Way Intelligent Adapter for the Microsoft Internet Information Server and how to configure the Web server components needed. A list of the files and directories created by the installation is also provided.

2.1 Installing the MS IIS e*Way

This section describes the procedure for installing the MS IIS e*Way.

Installation Procedure

If you are installing this e*Way as part of a complete e*Gate installation, please follow the instructions in the *e*Gate Integrator Installation Guide*. The MS IIS e*Way is installed as an "Add-on" component in the fourth phase of the installation.

If you are adding the MS IIS e*Way to an existing e*Gate installation, follow the instructions below.

2.2 Windows NT and 2000

Before installing e*Gate on your Windows NT system, read the sections in this chapter, to ensure a smooth and error-free installation.

Note: *You must have Administrator privileges to successfully install e*Gate on a Windows system.*

2.2.1 Pre-installation

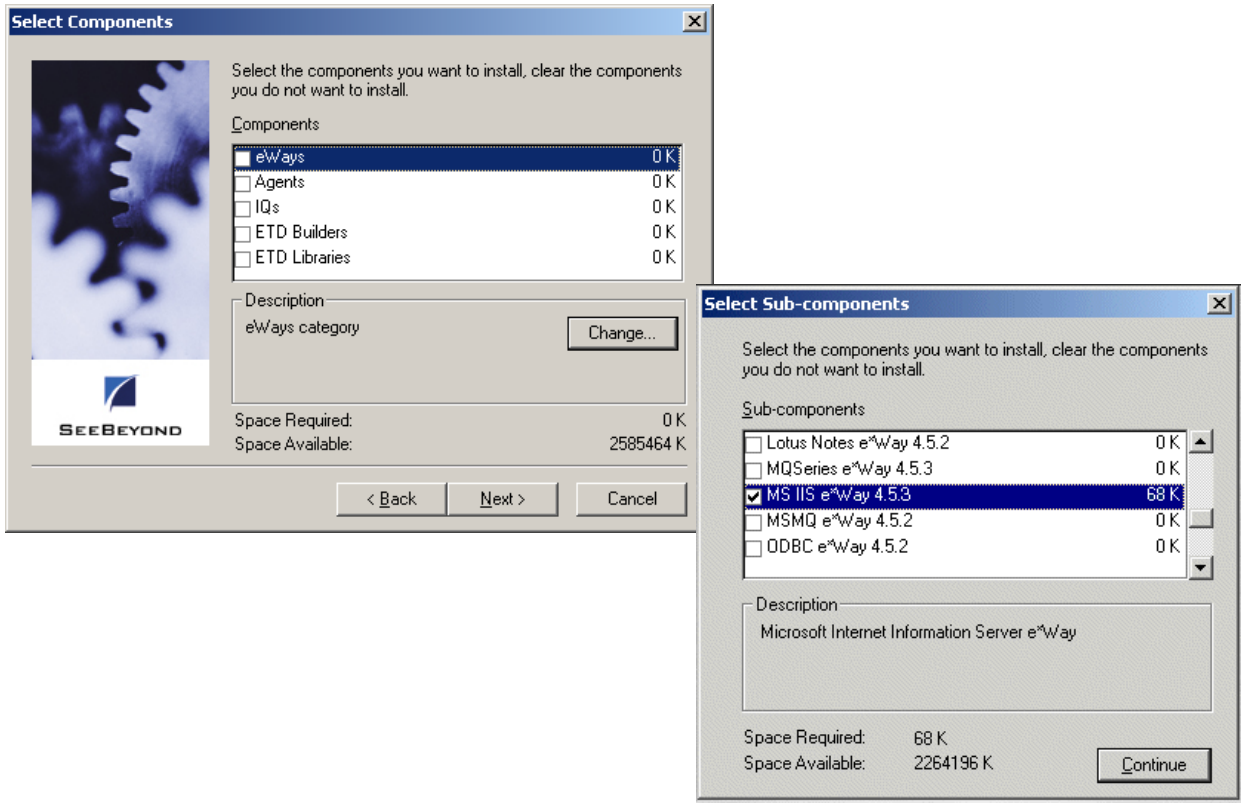
Before installation, take the following steps:

- Exit all programs before running the setup program, including any anti-virus applications.
- You must have Administrator privileges to install this e*Way.

To install the MS IIS e*Way on a Windows system:

1. Log in as an Administrator on the workstation on which you want to install the e*Way.
2. Close any open applications.
3. Launch the setup application on the e*Gate installation CD-ROM.
4. Follow the online prompts in the InstallShield® Wizard. When the **Select Components** dialog box appears, clear all the check boxes except **Add-ons**. Click **Next** as necessary to proceed through the setup application.
5. When the **User Information** dialog box appears, type your name and company name.
6. When the **Choose Destination Location** window appears, **do not** change the **Default Destination** folder unless you are directed to do so by SeeBeyond support personnel; simply click **Next** to continue.
7. When the **Select Components** dialog box appears, select e*Ways, click the **Change** button, and select **MS IIS e*Way 4.5.3.**

Figure 2 Add-ons Components



8. After the installation is complete, reboot the computer and run the e*Gate Enterprise Manager.

Note: *For details on how to use the Enterprise Manager graphical user interface (GUI), see the* ***e*Gate Integrator User's Guide.***

2.3 Files/Directories Created by Installation

The MS IIS e*Way installation process installs the files listed in Table 1 within the e*Gate directory tree. Files are installed within the **eGate\client** tree on the Participating Host and committed to the "default" schema on the Registry Host.

Table 1 lists the files installed on the server side.

Table 1 Server-side Files Installed

e*Gate Directory	Files
eGate\server\registry\repository\default\bin\win32	stc_iisapi.dll
eGate\server\registry\repository\default\external\ewmsisapi	stc_iisapi.properties

Note: *The target directories listed in this section are all suggested directories. If your knowledge of IIS permits and/or your needs require, you can use different directories.*

Table 2 lists the files that must be manually copied onto the system running the Web server.

Table 2 Client-side Files To Copy

Source Directories	Target Directories (IIS)	Files
\eGate\client\bin	\inetpub\msisapiext	stc_msclient.dll stc_mscommon.dll stc_msapi.dll stc_iisapi.dll
\eGate\client\external\ewmsisapi	\inetpub\msisapiext	stc_iisapi.properties

Table 3 lists the sample files that must be copied from the Installation CD-ROM manually to the Web server scripts directory.

Table 3 Sample Files To Copy

Installation CD-ROM	Files
\samples\ewmsisapi	readme.txt testmulptfrm.html testurlencoded.html webETD_MSAPI.zip

The sample files do *not* install automatically. They must be copied from the e*Gate installation CD-ROM to a temporary location.

Important: *The **stc_iisapi.properties** file is required to run the e*Way.*

See **"Configuring the Web Server Components" on page 18** and **"The Request/Reply Sample" on page 25** for more information.

Note: *After installation, be sure to change the file permission to allow the Web server to read and execute these files.*

Chapter 3

Configuration

This chapter explains how to configure the e*Way Intelligent Adapter for the Microsoft Internet Information Server.

3.1 Configuring Participating Host Components

This section explains how to configure the Participating Host components using the e*Gate Enterprise Manager graphical user interface (GUI).

Important

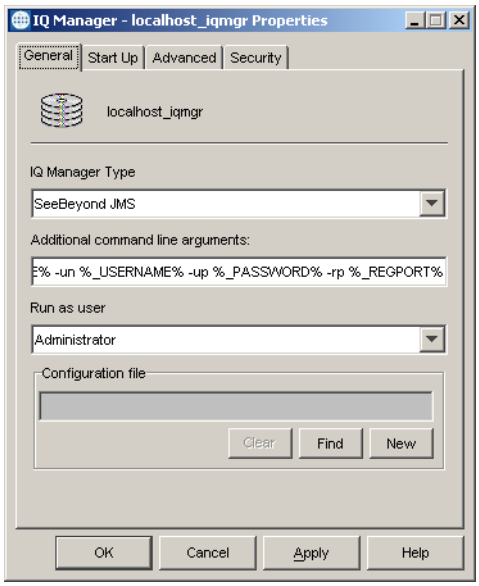
From the perspective of the e*Gate GUIs, the MS IIS e*Way is not a system of components distributed between the Web server and a Participating Host, but a single component that runs a dynamic link library file (the **stc_iisapi.dll**).

When this guide discusses procedures within the context of any e*Gate GUI, the term "e*Way" refers only to the Participating Host component of the MS IIS e*Way system.

To configure the Participating Host components:

1. If you have not already done so, launch the Enterprise Manager.
2. Verify that the Intelligent Queue (IQ) Manager Type is set to **SeeBeyond JMS** by double-clicking on the IQ Manager to view the associated properties.

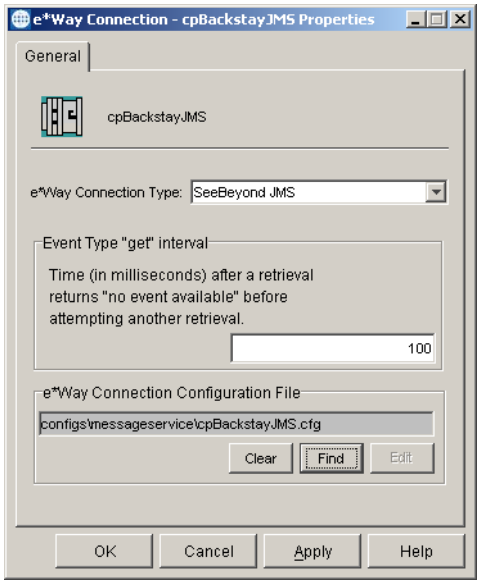
Figure 3 JMS IQ Manager



The **stc_iisapi.dll** publishes Events to the SeeBeyond Java Messaging Service (JMS), so the IQ Manager type in your Participating Host must be set to **SeeBeyond JMS** (the default).

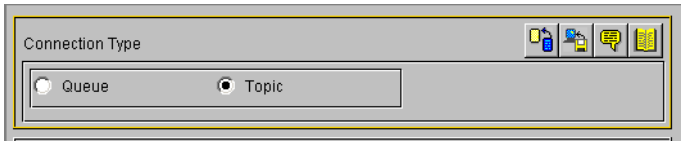
3 Create and configure an e*Way Connection. The connection type must be set to **SeeBeyond JMS** (for the sample, the e*Way Connection is **cpBackstayJMS**).

Figure 4 e*Way Connection



In the configuration file, set the connection type to **Topic**, because the sample is using the publish/subscribe model.

Figure 5 Connection Type



The server name is the machine on which your JMS server (JMS IQ Manager) is running. This is also the machine on which the Participating Host is installed. The host name is the same as the server name/IQ Manager.

4 Using the Component editor, create a new e*Way.

5 Display the new e*Way's properties. The default e*Way executable is **stceway.exe**.

6 Click **OK** to close the properties sheet, or continue to configure the e*Way. Configuration parameters are discussed in **Chapter 3**. The setup and requirements of schemas required to use this e*Way are discussed in **Chapter 4**.

7 Each e*Way has a Collaboration associated with it. Each Collaboration must have an associated source/destination.

Note: *Once you have installed and configured this e*Way, you must incorporate it into a schema by defining and associating the appropriate Collaborations, Collaboration Rules, IQs, and Event Types before this e*Way can perform its intended functions. For more information about any of these procedures, please see the online Help system.*

*For more information about configuring e*Ways or how to use the e*Way Editor, see the* ***e*Gate Integrator User's Guide*** *guide.*

3.1.1 Multi-Mode e*Way Configuration Parameters

Multi-Mode e*Way configuration parameters are set using the e*Way Editor.

To change e*Way configuration parameters:

1 In the Enterprise Manager's Component editor, select the e*Way you want to configure and display its properties.

2 Under **Configuration File**, click **New** to create a new file, **Find** to select an existing configuration file, or **Edit** to edit the currently selected file.

3 In the **Additional Command Line Arguments** box, type any additional command line arguments that the e*Way may require, taking care to insert them *at the end* of the existing command-line string. Be careful not to change any of the default arguments unless you have a specific need to do so.

For more information about how to use the e*Way Editor, see the e*Way Editor's online Help or the *e*Gate Integrator User's Guide*.

For more information about the Multi-Mode e*Way, see the *Standard e*Way Intelligent Adapter User's Guide*.

Configure the e*Way as needed for your system.

3.2 Configuring the Web Server Components

Each Web server requires different configuration. Consult your Web server documentation for complete information.

3.2.1 Configuring the IIS Web Server

The Web server loads and executes the client file, **stc_iisapi.dll** when a request arrives. It also needs to set the dynamic-load library path in order for **stc_msapi.dll, stc_mscommon.dll,** and **stc_msclient.dll** to be loaded by **stc_iisapi.dll**.

To configure the Web server to use the MS IIS e*Way Web server components via IIS Internet Services Manager:

1 Create a new virtual directory, such as **C:\Inetpub\msisapiext**.

If the default IIS server installation was used, the root directory is:

```
\inetpub
```

For example, create the following directory:

```
\inetpub\msisapiext
```

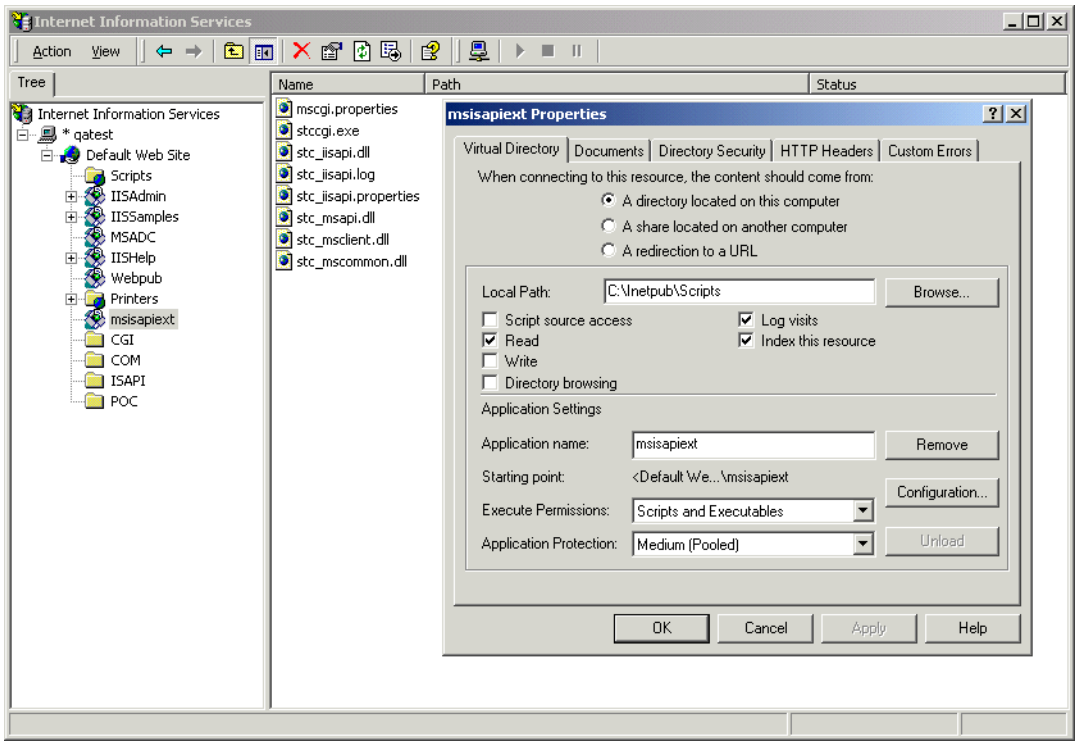
Note: *The creation of this directory is not mandatory but is recommended for easy maintenance, understanding and conformity to common industry practices. It is a good idea to create* ***msisapiext*** *to store all applications.*

2 Copy **stc_msapi.dll**, **stc_mscommon.dll**, **stc_iisapi.dll** and **stc_msclient.dll** to the new directory.

Note: *See* **Table 2 on page 13** *for a list of files and source directories.*

Figure 6 on page 19 shows an example of the IIS Virtual Directory.

Figure 6 IIS Virtual Directory



If you rename the **stc_iisapi.dll** to **yourapp.dll**, rename the **stc_iisapi.properties** file to **yourapi.properties**. The log file must follow the same naming conventions, as either **stc_iisapi.log** or **yourapp.log** in the same directory.

3 The URL to access the **stc_iisapi.dll** is:

```
/msisapiext/stc_iisapi.dll
```

or

```
/directory_name/yourapp.dll
```

4 Create/copy a test.html file to :

```
C:\Inetpub\wwwroot
```

or the doc root that was configured for IIS server.

5 You must modify **stc_iisapi.properties** to configure the **stc_iisapi.dll**. Change the permission on **stc_msapi.dll**, **stc_iisapi.dll**, **stc_msclient.dll** and **stc_mscommon.dll**, to enable the Web server to read and execute them.

For IIS, ensure that the directory created above, (**msisapiext**) Execute Permissions setting is set to **Scripts and Executables**. To modify this setting, go to Internet Service Manager, click on your Web site (for example, Default Web Site), right-click on **Scripts** and select **Properties**. In the Scripts Properties window, click on the **Virtual Directory** tab. **Select Scripts and Executables** on the **Execute Permissions** scroll menu. Click **OK** then restart the Web server.

Note: *Consult the Web server documentation for more information.*

6 To access the test*.**html** file from a Web browser, send a file to the MS IIS e*Way server. If operation is successful, you see the file you send to the server displayed. The URL to access the **stc_iisapi.dll** is:

```
http://hostname/msisiapiext/stc_iisapi.dll
```

A sample HTML form used to access **stc_iisapi.dll** follows:

```
<HTML>

<FORM ACTION="/msisiapiext/stc_iisapi.dll" METHOD="POST"
ENCTYPE="multipart/form-data">
Multipart test
<P>
<TABLE>
  <TR>
    <TD><LABEL for="fname">First name: </LABEL>
    <TD>  <INPUT type="text" name="firstname" id ="fname">
  <TR>
    <TD><LABEL for="lname">Last name: </LABEL>
    <TD><INPUT type="text" name="lastname" id="lname">
</TABLE>
    <LABEL for="email">email: </LABEL>
              <INPUT type="text" name="email"><BR>
    <INPUT type="radio" name="sex" value="Male"> Male<BR>
    <INPUT type="radio" name="sex" value="Female"> Female<BR>
    <LABEL for="filename">What files are you sending? </LABEL>
             <INPUT type="file" name="filename"><BR>
    <INPUT type="submit" value="Send"> <INPUT type="reset">
    </P>
</FORM>

</HTML>
```

3.2.2 Modifying the stc_iisapi.properties File

You must edit the **stc_iisapi.properties** file before running the MS IIS e*Way. This file contains the information pertaining to the JMS connection, data, and logging values.

This .**properties** file is loaded by the JMS **stc_iisapi.dll**. Each property is a name/value pairing. The name uniquely identifies the property. The value is the content associated with that name. The name is separated from the value with the ":" (colon) character.

Important: *Do **not** change the names. The .**properties** file is loaded only once, when the .**dll** file is loaded. If you change any of the property values, you must restart the IIS server to enable the changes.*

JMS Connection Section

Host

The name of the host where the JMS is running. The JMS IQ Manager acts as the message service (server). If the host name is not specified, **localhost** is the default value.

```
Host:localhost
```

Port

The port at which the JMS is listening for connections. If port is not specified, 24053 is the default value.

```
Port:24053
```

Timeout

Time-out for Request/Reply. This specifies the time-out in milliseconds to wait for the reply. The value entered here provides the time the back-end requires to process the message. This is used only for the Request/Reply mode.

```
Timeout:60000
```

DestinationType

Selects the JMS mode as Topic or Queue request. Specify "t" for Topic or "q" for Queue requests. Topic request is used for Topic publishing, meaning, one publisher, potentially multiple subscribers. Each subscriber gets the same copy of the message. Queue request is used for point-to-point, meaning one sender, with a possible group of receivers, where only one receiver actually receives the message. Use lower case.

```
DestinationType:t
```

DestinationName

If you are publishing to a topic, enter the topic name; if you are publishing to an IQ enter the IQ name.

```
DestinationName:etwebRequestETDTopic
```

RequestReply

Selects the JMS delivery mode as Request/Reply. Specify **True** for the Request/Reply mode (see **Timeout** for details on how to configure reply **Timeout**). Specify **False** for the Publish or Send modes only. In these cases, the system does not expect a JMS reply, but instead you get a generic reply saying the request is done.

```
RequestReply:True
```

MaximumConnections

Specifies the maximum number of JMS connections allowed in the e*Way Connection pooling. Any integer below 100 is acceptable. This value depends on the IIS server setup and can change. If the server can only handle 256 concurrent HTTP requests, you do not need to set this parameter to a higher value than 256.

```
MaximumConnection:100
```

ClientID

The Client ID to use for the JMS connection.

```
ClientID:SeeBeyondMSIIS
```

TimeToWaitForConnection

The time to wait in milliseconds for a JMS connection if all connection in the connection pool are used.

```
TimeToWaitForConnection:10000
```

MS IIS API Data Section

ReadChunksize

When the **stc_iisapi.dll** performs a ReadClient. ReadChunksize specifies the chunk size, in bytes, of data to be read.

If you specify 1024 then isapi will read 1024 bytes of data once a time. If the Control Block has all of the data, then isapi will not perform ReadClient, it just copies the data from the control block to the JMS message block. The default internal read chunk size is 409600 bytes. ReadChunksize is an integer value, the max you can specify is 2147483647 bytes.

```
ReadChunkSize:409600
```

WriteChunksize

When isapi does write to HTTP server, WriteChunksize specifies the chunk size in bytes, of the data to be written at one time. If you specify 1024 then isapi will write 1024 bytes of data once a time. The default internal write chunk size is 409600 bytes. WriteChunksize is an integer value, the max you can specify is 2147483647 bytes.

```
WriteChunkSize:409600
```

Log Section

Trace

The trace level to use for trace/debug. The following are valid values:

0	Info (in addition to all three categories below)
1	Warn (in addition to both categories below)
2	Error (in addition to the category below)
3	Fatal (only)

```
Trace:0
```

3.2.3 Hex Dump vs. Text Dump

The MS IIS e*Way Web server component provides *dumps*, that is, writes the contents of each message to the log file). The component writes the contents of every inbound (request) and outbound (reply) message that it handles, provided the Trace level in **stc_iisapi.properties** file is set to **0** (zero).

There are two types of dumps that can occur: a text dump and a hex dump. Text dumps are formatted into standard text. Hex dumps are formatted into lines of 16 bytes with two representations each, in per section. The first section is the hex representation of 16 bytes, followed by the second, which contains the ASCII representation of the same 16 bytes. If any byte is nonprintable, a dot is substituted. The type of dump that occurs is determined by the content type of the message.

If the inbound message to the MS IIS e*Way (the date read from the ReadClient) is any content-type other than text/*, a hex dump occurs. If the content-type is text/*, a text dump occurs. The inbound hex dump does not include any environment variables from the HTTP server, for example, CONTENT_TYPE, CONTENT_LENGTH, PATH, or HTTP_ACCEPT.

If the **Default Outgoing Message Type** parameter in the e*Way Connection configuration is set to publish "bytes" messages, a hex dump occurs. If it is set to publish "text" messages, a text dump occurs. See the following examples:

Sample Hex Dump

```
JMS  I 3872 (iisapi.cxx:1185):  2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2
D 2D 2D 2D  ----------------
JMS  I 3872 (iisapi.cxx:1185):  2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2
D 37 64 32  -------------7d2
JMS  I 3872 (iisapi.cxx:1185):  33 64 37 31 33 38 30 31 38 34 0D 0A 4
3 6F 6E 74  3d71380184..Cont
JMS  I 3872 (iisapi.cxx:1185):  65 6E 74 2D 44 69 73 70 6F 73 69 74 6
9 6F 6E 3A  ent-Disposition:
JMS  I 3872 (iisapi.cxx:1185):  20 66 6F 72 6D 2D 64 61 74 61 3B 20 6
E 61 6D 65   form-data; name
JMS  I 3872 (iisapi.cxx:1185):  3D 22 66 69 72 73 74 6E 61 6D 65 22 0
D 0A 0D 0A  ="firstname"....
JMS  I 3872 (iisapi.cxx:1185):  66 67 66 67 0D 0A 2D 2D 2D 2D 2D 2D 2
D 2D 2D 2D  fgfg..----------
```

Sample Text Dump

```
JMS  I 3872 (iisapi.cxx:1395): writeClient JMS TextMessage  of [6124]
 bytes actual written [6124] bytes
JMS  I 3872 (iisapi.cxx:1397): ------=_Part_0_1870449.1016041716682
Content-Type: text/plain; content-transfer-encoding=base64
Content-Disposition: form-data
Content-Description: add a sample description in collab
```

Zmdm

```
------=_Part_0_1870449.1016041716682
Content-Type: text/plain; content-transfer-encoding=base64
Content-Disposition: form-data
Content-Description: add a sample description in collab
```

Zmdm

```
------=_Part_0_1870449.1016041716682
Content-Type: text/plain; content-transfer-encoding=base64
Content-Disposition: form-data
Content-Description: add a sample description in collab
Zmdm
```

Chapter 4

Implementation

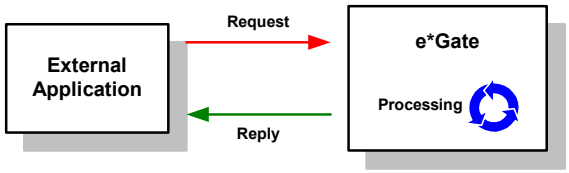
This chapter explains how to implement the e*Way Intelligent Adapter for the Microsoft Internet Information Server.

4.1 The Request/Reply Model

All the applications of the MS IIS e*Way are based upon the "Request/Reply" model (see Figure 7). At a high level, this model works as follows:

- Request/Reply, that is, data is sent to the e*Gate system and a response is returned
- Send-only or "fire and forget," that is, data is sent to the e*Gate system but no data is returned from the e*Gate system

Figure 7 The Request/Reply concept



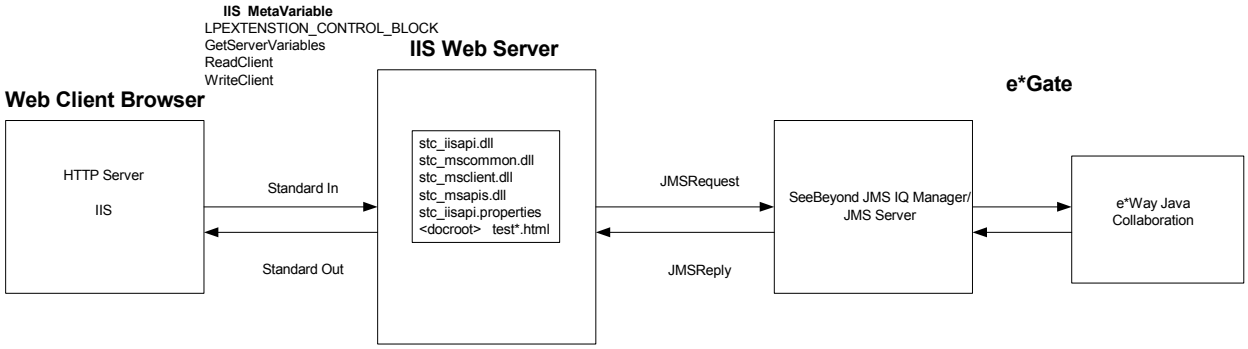
In the "fire and forget" model, the "extension" **stc_iisapi.dll** sends a simple reply on behalf of the IIS server to acknowledge the HTTP request (so the HTTP request does not time-out). Inside e*Gate, you do not need to reply to the Event.

4.1.1 Request/Reply and e*Way Participating Host Components

The MS IIS e*Way Participating Host component is a Multi-Mode e*Way that uses a proprietary IP-based protocol to multi-thread Event exchange between the e*Way and external systems or other e*Gate components.

Figure 8 on page 25 illustrates how the e*Way receives data from an external application and returns processed data to the same application.

Figure 8 Data flow



The operation shown in Figure 8 proceeds as follows:

1. The user accesses one of the test*.**html** files from a Web browser, sending a file to the MS IIS e*Way server, using either HTTP Post or Get.
2. A JMSRequest is made, the JMSReplyTo property is assigned, and a TemporaryTopic subscriber is created.
3. The JMSRequest is passed to the SeeBeyond Java Messaging Service (JMS) server, that is, the JMS Intelligent Queue (IQ) Manager, which then uses the etwebRequestETDTopic Event Type name to forward the text message to an e*Way Collaboration.
4. Collaborations within the e*Way perform any appropriate processing that may be required, and route the processed Events to other destinations (such as an external system for additional data retrieval or processing, then back to the Web server as a Reply, using TemporaryTopic.
5. The Web server gets the content from TemporaryTopic, and replies to the Web client.

4.1.2 The Request/Reply Sample

The sample schema can be found in the e*Gate installation CD-ROM in the **samples/ewmsisapi** directory.

e*Gate Request/Reply Sample Set up

Request Reply Sample

1. Install the MS IIS e*Way Server add on.
2. Import the sample schema

eGate 4.5.1 and higher

3. In eGate Enterprise Manager, go to the File menu and select "Import Definitions from File", select Schema in the Import wizard, select webETD_MSAPI.zip for Schema File Name. RequestReply.zip is the file supplied in this sample directory.

These instructions also appear in the "readme" files in the MS IIS e*Way samples directory (**samples/ewmsisapi**).

Once the client and server are set up, you can test the entire system using a Web browser:

1. Start the Control Broker. This also starts the JMS server (JMS IQ Manager).
2. Start the desired e*Way.

   In the sample each e*Way demonstrates different functionality.
3. Use a browser to open the **test*.html** file.
4. Fill out and submit the form.
5. Confirm that the **stc_iisapi.dll** returned the form data as expected. You can view the **msisapi.log** file, located in **msisapiext**, to see the content sent out/received by **stc_iisapi.dll** from the JMS server.

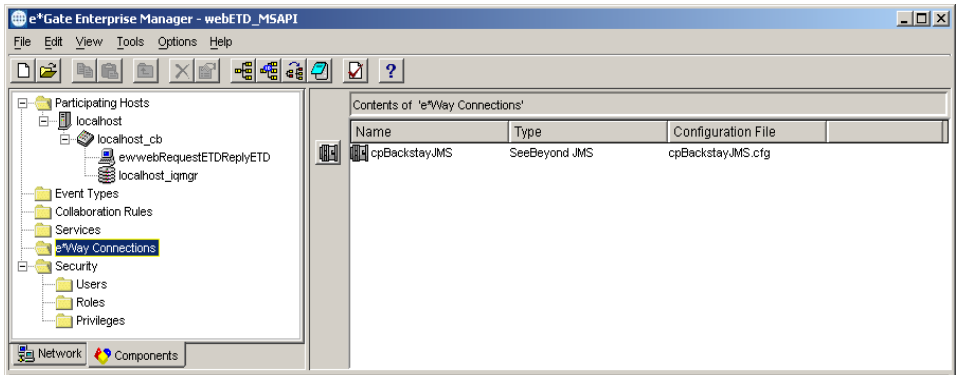
Sample Functionality

There are separate e*Ways that handle various activities. There are four basic steps involved in a standard Request/Reply schema:

1. Define the content.
2. Populate the webReplyETD payload.
3. Call the send method in the webReplyETD.
4. Using the JMSReplyTo property, send the input to the webRequestETD.

Once successfully installed, the Enterprise Manager will open to the following:

Figure 9 webETD_MSAPI Enterprise Manager Components View



The sample is comprised of one e*Way:

- ewwebRequestETDReplyETD

ewwebRequestETDReplyETD

The ewwebRequestETDReplyETD e*Way receives a block of data, parses the input into webRequestETD, allows you to create webReplyETD, marshalling the data into a block, and sends the block of data back.

crRequestReply_webRequestETDReplyETD

The Collaboration Rule associated with the ewwebRequestETDReplyETD e*Way performs the following:

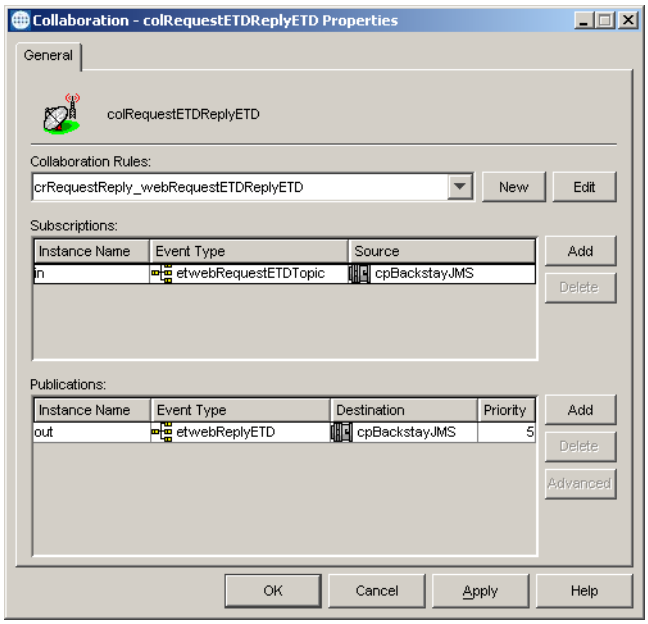
1. Provides the HTTP content type for the reply message.
2. Gets the specified HTTP headers that **isapi** passes back to the Web server.
3. Demonstrates how to get the decoded inbound payload.
4. Demonstrates how to set outbound encoding. Encoding is not required, as it will be performed automatically.
5. Shows charSet encoding/decoding activity. You perform a get of the decoded charSet payload from textStringPayLoad. If you require outbound encoding of the charSet, set the outbound charSet attributes.
6. Demonstrates how to get multiPart payload, how to drill down on nested multiPart payload.
7. Demonstrates how to build an outbound multiPart payload.
8. Demonstrates how to get HTTP_GET URLencoded name value pairs. The URLencoded string is decoded into name value pairs.
9. Sends the HTTP-enabled data to the specified reply topic.

colRequestETDReplyETD

The Collaboration that ties the e*Way and the Collaboration Rule together is "colRequestETDReplyETD". The Collaboration must have an Event Type, and Source/Destination defined for both subscription and publication. For colRequestETDReplyETD Collaboration, the Collaboration subscribes to etWebRequestETDTopic Event Type. Remember, that you have configured **stc_iisapi.dll** to publish etWebRequestETDTopic, this is where you see the publication and subscription of the topic. The temporary topic acts replies to the request, and is therefore, transparent to the user. The exact temporary topic is stored in the input Event JMSReplyTo node.

The Collaborations subscribe and publish to the same external source/destination "cpBackstaryJMS" (the e*Way Connection).

Figure 10 colRequestETDReplyETD



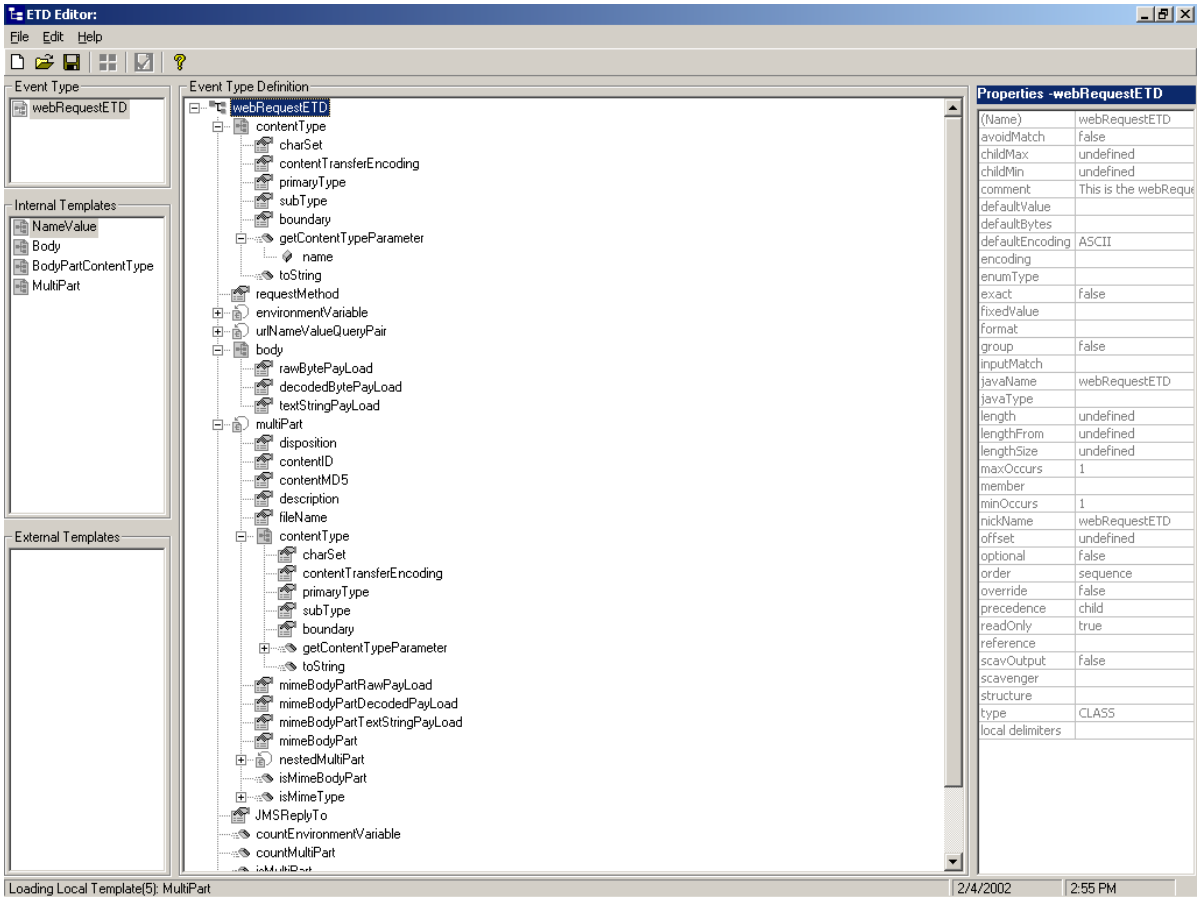
4.1.3 Event Type Definitions

As discussed in **Request/Reply and e*Way Participating Host Components** on page 24, the MS IIS e*Way maintains "JMSReplyTo" property that must be set to facilitate the reply data flows through the e*Gate system.

webRequestETD

The webRequestETD Event Type Definition (ETD) that can be used within a Request/Reply schema is shown below. The webRequestETD is designed to be "read" only, while the webReplyETD is designed to be "write" only.

Figure 11 The webRequestETD



Node Descriptions

contentType

The top level node contentType represents the CONTENT_TYPE meta variable as in isapi. If the message can be defined as a discrete-type media, such as text/*, image/*, the content type shows up at the top level.

The following subnodes appear below it.

charSet

If the charSet is not supported, or misspelled, the decoding can not take place. In that case, use rawBytePayLoad, or decodedBytePayLoad. The charSet encoding table is available at:

http://www.ingrid.org/java/i18n/encoding/table.html

The value can be null if the message does not have the charset attribute.

contentTransferEncoding
Indicates whether the content is encoded (base64, binary etc.) the field is not case sensitive. It may be return a null string if not transfer encoded.

primaryType
Indicates the primary content type. For example, text in text/xml.

subType
Indicates the sub type of the content. For example, xml in text/xml.

boundary
If it is multipart/* data, this node carries the value of the boundary. If the message is nested multipart, this will simply provide the top level boundary.

getContentTypeParameter
Provides a method to be called to get any other content type attributes, such as ContentID, may also return a null string, if the attribute is not defined.

requestMethod
Returns the HTTP method. For example, POST or GET.

environmentVariables

A top level node that contains an array of isapi variables received or passed to **msisapi**. For example HTTP_ACCEPT.

name
Return the name of the environment variable.

value
Returns the value of the environment variable.

urlNameValueQueryPair

A top level node that contains an array. Check the count of this array, if it returns a value equal to zero, the content is not a URL encoded value pair. If HTTP server received a get command, a URL encoded query string is received. The ETD decodes the string, and populates the name value pair for use directly.

name
The name of the part. For example,

```
firstname
```

value
The value of the part. For example,

```
myfirstname
```

body

A top level node, that will contain the body of the content for discrete types, such as text/* or image/*.

The following subnodes appear below it.

rawBytePayLoad
Contains raw byte data.

decodedBytePayLoad

If the content indicates that it has been transfer encoded in the contentType header, content-transfer-encoding attribute, decodes the rawBytePayload. The result is the decoded byte array, while rawBytePayLoad still contains the original encoded byte data.

textStringPayLoad

If the content primary type indicates that it is text based, a string is created from rawBytePayLoad (or decodedBytePayLoad, if the content has been transfer encoded). The resulting text string is a Java internal representation of the original text string, in the designated charSet character encoding.

For example, a byte stream of data is received that is EUC_JP character encoded, the textStringPayLoad produces a Java internal string representation (Unicode) of that Japanese character stream. It is not in EUC_JP encoding, but in Java Unicode.

If the charSet is not recognized by the java decoder, (for example, you misspelled EUC_JP to ECU_JP) then you will not be provided with a textStringPayLoad. You can access the data from rawBytePayLoad or decodedBytePayLoad.

If the original message contains a content transfer encoded text string, the rawBytePayLoad, decodedBytePayLoad, and textStringPayLoad will all contain data.

If you specify the content type as application/xml, rather than text/xml, this media form is not recognized as a text type. textStringPayLoad is not populated, even though the content body is a text based byte array. In this case the content must be retrieved from rawBytePayLoad or decodedBytePayLoad (if content transfer encoded).

If the content type is not text/* and is not multipart/*, the payload must be retrieved via rawBytePayLoad or decodedBytePayLoad (if content transfer encoded).

For example, to use the Content-Type: text/plain;Content-Transfer-Encoding="Quoted-Printable" with the text "Now is the time for all folk to come to the aid of msisapieway," the quoted-printable transfer encoding will appear as below:

```
Now is the time=
For all folk to come=
To the aid of msisapieway=
```

For more information, refer to RFC2045, (page 16) at **http://www.ietf.org/rfc/rfc2045.txt**.

Per RFC2045

```
Content-Type:text/plain; charset=EUC_JP
Content-transfer-encoding: base64
```

In accordance with RFC2045, the body is a base64 US_ASCI encoding of data that was originally, in EUC_JP. After unmarshalling via webRequestETD, the textStringPayLoad provides a Java string Unicode encoding of the Japanese content originally in EUC_JP.

multiPart

A top level node, used in conjunction with multipart/* content type data, the following subnodes are available:

disposition
The content-disposition of each body part in the multipart array. Check for null values. For example, multipart/form data will have top level content type populated as multipart/form:boundary=____. Each part has disposition form data.

contentID
Contains the ContentID attributes, check for null values.

contentMD5
Returns the value of the "content-MD5" header field. Returns null if this field is unavailable, or absent. MD5 is a 128 bit digital finger print.

description
Returns the "content-Description" field.

filename
If the content contains a filename, returns the value, otherwise, null.

contentType
This allows you to access all contentType parameters. Similar to the top level contentType node, this contentType indicates the break down of attributes for this body part. Check for null values.

mimeBodyPartRawPayLoad
Contains the raw byte data.

mimeBodyPartDecodedPayLoad
If the content indicates that it has been transfer encoded in the contentType header, content-transfer-encoding attribute, decodes the rawBytePayLoad, returning a decoded byte array. In this case the mimeBodyPartRawPayLoad still contains the encoded byte data.

mimeBodyPartTextStringPayLoad
If the content primary type indicates that it is text data, a string is created from the mimeBodyPartRawPayLoad (or the mimeBodyPartDecodedPayLoad if the content is transfer encoded). The resulting text string is a Java internal representation of the original text string, in the designated charSet character encoding.

For example, a byte stream of data is received that is EUC_JP character encoded, the textStringPayLoad produces a Java internal string representation (Unicode) of that Japanese character stream. It is not in EUC_JP encoding, but in Java Unicode.

If the original message contains a content transfer encoded text string, the mimeBodyPartRawBytePayLoad, mimeBodyPartDecodedBytePayLoad, and mimeBodyPartTextStringPayLoad will all contain data.

If you specify the content type as application/xml, rather than text/xml, this media form is not recognized as a text type. mimeBodyPartTextStringPayLoad is not populated, even though the content body is a text based byte array. In this case the content must be retrieved from mimeBodyPartRawBytePayLoad or mimeBodyPartDecodedBytePayLoad (if content transfer encoded).

If the content type is not text/* and is not multipart/*, the payload must be retrieved via mimeBodyPartRawBytePayLoad or mimeBodyPartDecodedBytePayLoad (if content transfer encoded).

For example, to use the Content-Type: text/plain;Content-Transfer-Encoding="Quoted-Printable" with the text "Now is the time for all folk to come to the aid of msisapieway", the quoted-printable transfer encoding will appear as below:

```
Now is the time=
For all folk to come=
To the aid of msisapieway=
```

For more information, refer to RFC2045, (page 16) at **http://www.ietf.org/rfc/rfc2045.txt**.

Per RFC2045

```
Content-Type:text/plain; charset=EUC_JP
Content-transfer-encoding: base64
```

In accordance with RFC2045, the body is a base64 US_ASCI encoding of data that was originally, in EUC_JP. After unmarshalling via webRequestETD, the `textStringPayLoad` provides a Java string Unicode encoding of the Japanese content originally in EUC_JP.

mimeBodyPart
Returns an object of `javax.Mail.Internet.MimeBodyPart`. This value can be accessed via multipart nesting. This provides the means to access a raw object handle.

nestedMultiPart
If the multiPart contains another multipart, as an associated body part, this node allows for drilling down to the nested multiPart. If it is not nested, it returns null. The `mimeBodyPartRawPayLoad` and `decodedPayLoad` are maintained at the same time.

isMimeBodyPart
Queries whether the body part is of a mime type.

isMimeType
Queries whether the body part is of a certain mime type.

JMSReplyTo

Gets the JMSReplyTo property. See**"Event Type Definitions" on page 28** for more information about JMSReplyTo properties.

countEnvironmentVariable

Returns an integer, indicating the number of environment variables.

countMultiparts

Returns an integer, indicating the number of body parts.

isMulitpart

Returns true or false, indicating whether multipart or not.

isUrlencoded

Returns true or false, indicating whether the data is URL encoded.

isSingleBody

Returns true or false, indicating whether the data consists of a single body part.

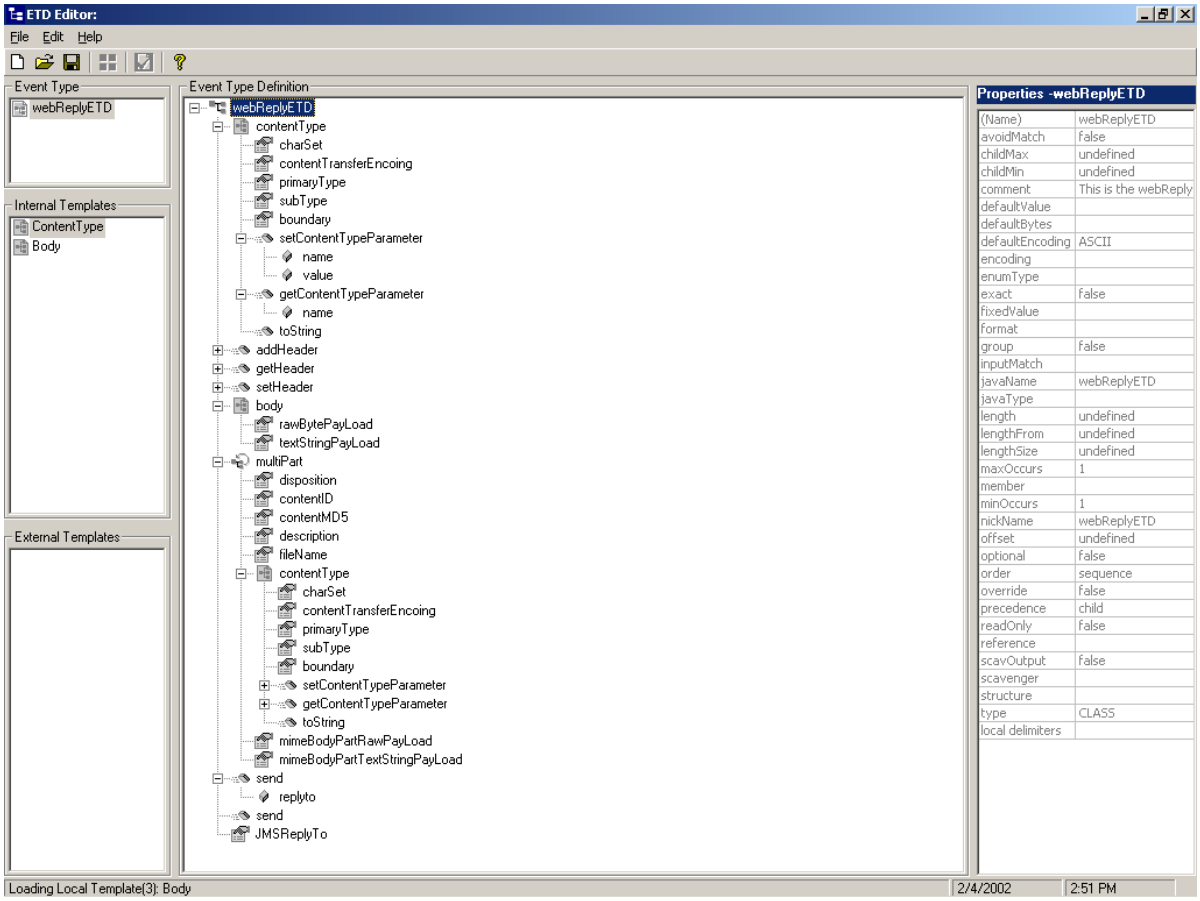
countURLNameValueQueryPairs

Returns an integer, indicating the number of name/value pairs.

webReplyETD

The webReplyETD Event Type Definition (ETD) that can be used within a Request/Reply schema is shown in the following sections. The webReplyETD mirrors the webRequestETD. The webReplyETD is "write" only, while the webRequestETD is "read" only. There are functions availabe to allow you to read the node value, but the node value must be set, before reading.

Figure 12 The webReplyETD



Node Description

contentType

The CONTENT_TYPE header returned to the HTTP client. This contentType is prepended to the body, with two CRLF (carriage return line feed), and the body.

The following subnodes appear below it.

charSet
Setting this value, causes a byte array of text data to result as the output. You must also then set the Body node's textStringPayLoad.

contentTransferEncoding
Setting this value, defines the content as encoded (such as base64). If the data is multipart, do not set transfer encoding at the top level.

primaryType
Sets the primary content type. For example, text in text/xml.

subType
Sets the sub type of the content. For example, xml in text/xml.

boundary
This value is only set if you do not want to use the system defined boundary. Do not set if the data is not multipart data.

setContentTypeParameter
Sets a name/value pair in contentType.

getContentTypeParameter
Gets a name/value pair from contentType.

addHeader

A method, that allows you to add other http/isapi response headers, taking a name/value pair as the parameters. You do not need to add Content_Type headers here, since that is already accomplished by the setting the contentType node. The ETD bundles the contentType headers behind the scenes.

getHeader

A method, that allows you to retrieve the header, taking a name/delim pair as the parameters.

setHeader

A method, that allows you to set the header value, taking a name/value pair as the parameters.

body

A top level node, if the data is of the discrete type, such as text/* or image/*, the body of content appears here. You either set rawBytePayLoad or textStringPayLoad, not both.

The following subnodes appear below:

rawBytePayLoad
If the data is not text based, the raw byte data is contained here.

textStringPayLoad

Set this pay load for text string based data. If this value is set, the content type is not verified. The text string can be from any language, as specified by :

http://www.ingrid.org/java/i18n/encoding/table.html

You need to specify the out put charSet, if the string contains any non-default text, such as EUC_JP, SJIS, GB2312, ISO-8859-1. Even if you do not want to perform any conversion, such as EUC_JP to SJIS, specify the desired output character set. For example, if the textStringPayLoad contains java Unicode string EUC_JP, specify charSet=EUC_JP. The Unicode output results in a byte array that contains EUC_JP characters. If you set charSet=SJIS, the output results in Java Unicode encoding of the original EUC_JP string to a byte array that contains an SJIS character set.

multiPart

This is an array of body part. The ETD allocates a new body part for you, the first time you access the body part attribute, while you are responsible to set each body part pay load. If the boundary is set in the contentType node, that boundary value is used. If you only set the header and neglect to set the payLoad, you will receive a marshalling exception.

The following subnodes appear below:

disposition

Sets the disposition.

contentID

Sets the contentID.

contentMD5

Sets the MD5 signature.

description

Sets the description.

filename

Set the filename.

contentType

Allows you to set the values of the following subnodes:

charSet

Setting this value, causes a byte ar ray of text data to result as the output. You must also then set the Body node's textStringPayLoad.

contentTransferEncoding

Setting this value, cause a byte array of text data to result as the output. You must also then set the body node's textStringPayLoad.

primaryType

Sets the primary content type. For example, text in text/xml.

subType

Set the sub type of the content. For example, xml in text/xml.

boundary
This value is only set if you do not want to use the system defined boundary. Do not set if the data is not multipart.

setContentTypeParameter
Sets a name/value pair in contentType.

getContentTypeParameter
Gets a name/value pair from contentType.

mimeBodyPartRawPayLoad
Sets the body part payload to "raw".

mimeBodyPartDecodedPayLoad
Sets the body part payload to "decoded".

send

Sends the reply to the JMSReply property, taking the replyTo parameter string.

send

Sends the reply to the JMSReply property.

JMSReplyTo

Sets the JMSReplyTo property.

4.1.4 Collaboration Rules and the IIS Header

The Collaboration Rule provided in the sample (shown in Figure 13) demonstrates a variety of possible behavior.

1. Provide the HTTP content type for the reply/outbound message:

```
getout().getContentType().setPrimaryType("text")
getout().getContentType().setSubType("plain")
```

2. Get request/inbound isapi meta variables:

```
getin().getEnvironmentVariables(i).getName()
getin().getEnvironmentVariables(i).getValue()
```

3. Add another reply/outbound HTTP header:

```
getout().addHeader(getin().getEnvironmentVariables(i).getName(),ge
tin().getEnvironmentVariables(i).getValue())
```

4. Test/Get decoded inbound payload:

   Multipart test:

```
getin().getMultiParts(j).getContentType().getContentTransferEncodi
ng()==null
```

   Single body:test

```
getin().getContentType().getContentTransferEncoding()==null
```

   Multipart get decoded payload:

```
byt[]temppayload=getin().getMultiParts(outmlptindex).getMimeBodyPa
rtDecodedPayLoad()
```

   Single body decoded payload:

```
byte[]bytesinglebody=getin().getBody().getTransferCodePayLoad()
```

If it is text message, it is always decoded for you with:

```
String strsinglebody=getin().getBody().getTextStringPayLoad()
```

5 Set charSet or content transfer encoding for reply/outbound, set proper payload, the encoding is performed for you:

```
getout().getContentType().setCharSet("SJIS")
getout().getMultiParts(outmlptindex).getContentType().setContentTr
ansferEncoding("base64")
getout().getMultiParts(outmlptindex).setMimebodyPartTextStringPayL
oad(getin().getMultiParts(j).getMimeBodyPartTextStringPayLoad())
getout().getMultiParts(outmlptindex).setMimeBodyPartRawPayLoad(tem
ppayload)
```

6 How to get URL encoded name/value pair:

```
getin().isUrlencoded()
strbuf4.append(" url name:
"+getin().getURLNameValueQueryPairs(k).getName()+" value:
"+getin().getURLNameValueQueryPairs(k).getValue())
```

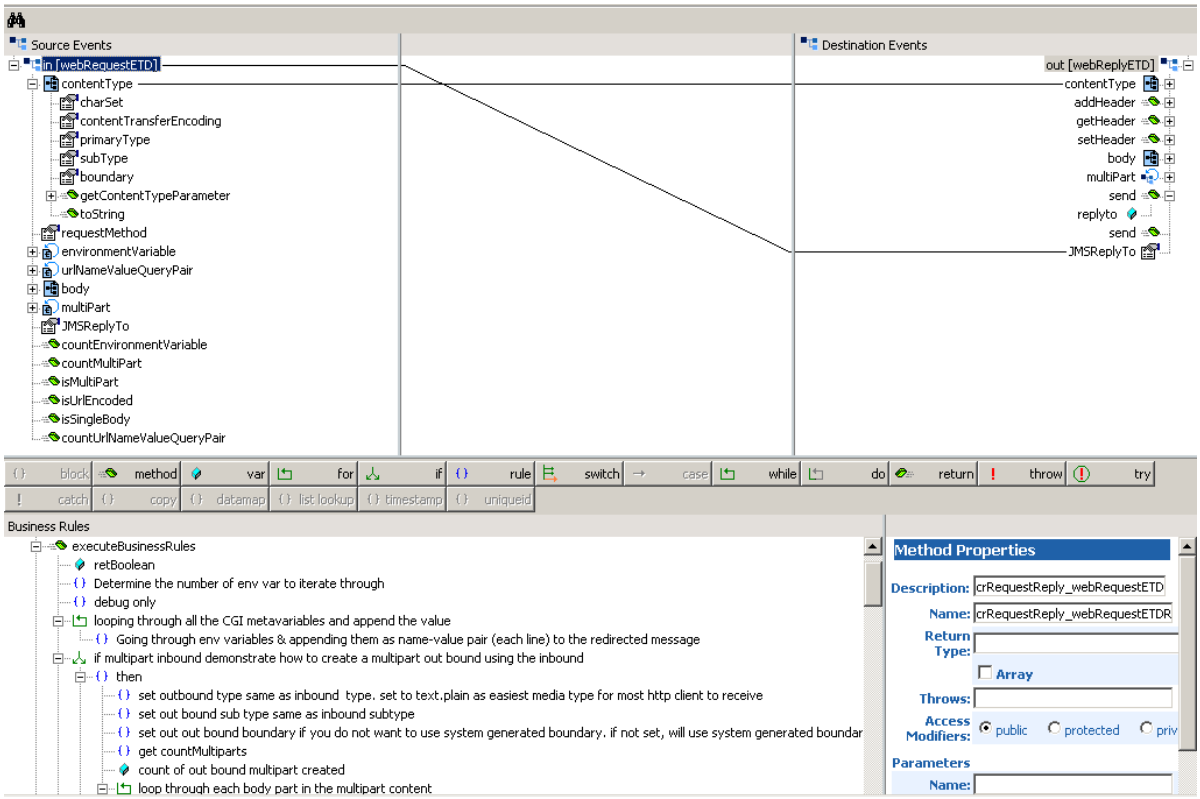
7 Send the HTTP-enabled data to the reply topic:

Manual publish:

```
getout().send(getout().getJMSReplyTo())
```

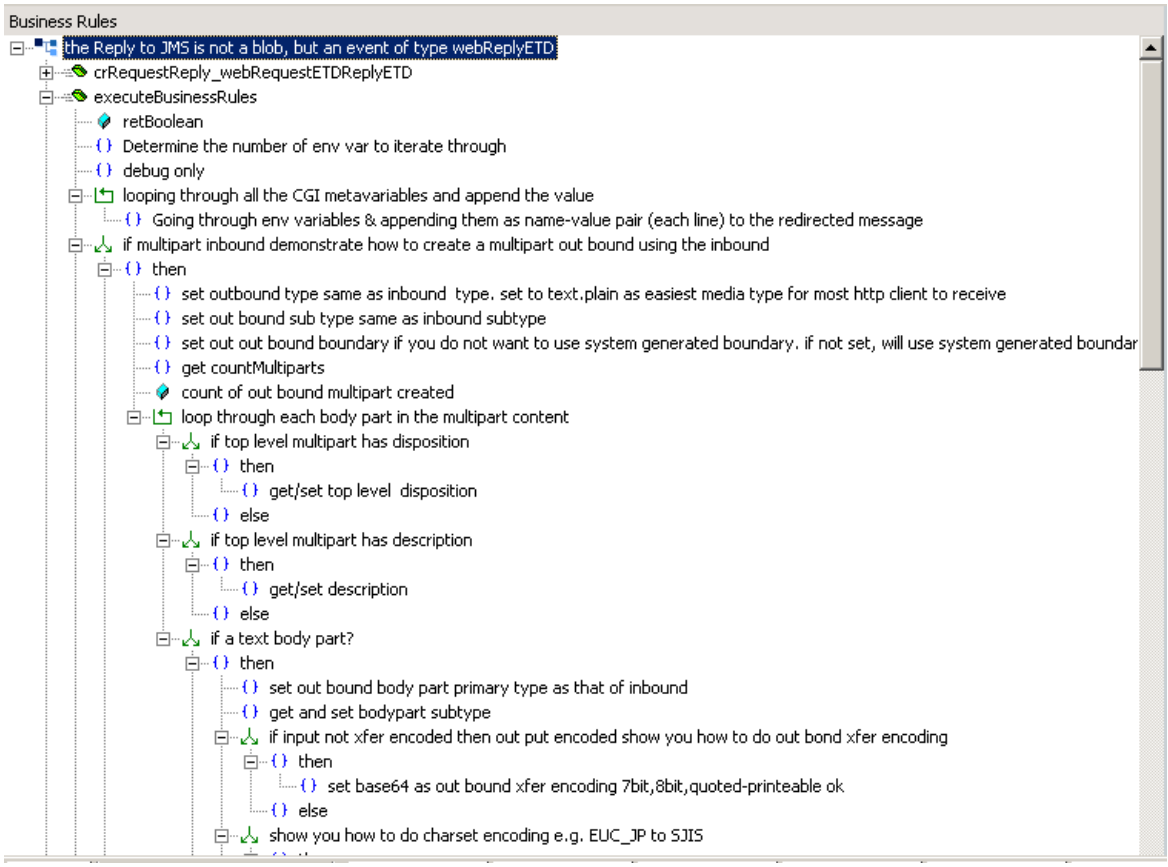
crRequestReply_webRequestETDReplyETD

Figure 13 Copying the Request/Reply field



A portion of the Business Rules that were implemented for crRequestReply_webRequestETDReplyETD Collaboration Rules sample, appear below:

Figure 14 Business Rules



The code itself can be viewed in the GUI from the View menu, select View Java Code. The code appears as follows:

```
import com.stc.common.collabService.*;
import com.stc.jcsre.*;
import com.stc.eways.util.*;
import java.io.*;
import java.sql.*;
import java.util.*;
import com.stc.eways.webETD.*;


   class crRequestReply_webRequestETDReplyETDBase extends JCollaborat
ion
   {
      public crRequestReply_webRequestETDReplyETDBase()
      {
         super();
      }

      com.stc.eways.webETD.webRequestETD in = null;

      public com.stc.eways.webETD.webRequestETD getin()
```

```
      {
         return this.in;
      }

      com.stc.eways.webETD.webReplyETD out = null;

      public com.stc.eways.webETD.webReplyETD getout()
      {
         return this.out;
      }

      public void resetData() throws CollabConnException, CollabDataE
xception
      {
         this.in = (com.stc.eways.webETD.webRequestETD) this.reset((E
TD)this.getin());
         this.out = (com.stc.eways.webETD.webReplyETD) this.reset((ET
D)this.getout());
      }
      public void createInstances() throws CollabConnException
      {
         this.in = (com.stc.eways.webETD.webRequestETD) this.newInsta
nce("com.stc.eways.webETD.webRequestETD", "in", ETD.IN_MODE);
         this.out = (com.stc.eways.webETD.webReplyETD) this.newInstan
ce("com.stc.eways.webETD.webReplyETD", "out", ETD.OUT_MODE);
      }
   }



   public class crRequestReply_webRequestETDReplyETD extends crReques
tReply_webRequestETDReplyETDBase implements JCollaboratorExt
   {
      public  crRequestReply_webRequestETDReplyETD()
      {
         super();
      }

      public boolean executeBusinessRules() throws Exception
      {
         boolean retBoolean = true;
         int EnvVarCnt = getin().countEnvironmentVariables();
         System.err.println("boundary:"+getin().getContentType().getM
ultipartBoundary());
         for( int i = 0;i < EnvVarCnt;i++)
         {
            getout().addHeader(getin().getEnvironmentVariables(i).get
Name(),getin().getEnvironmentVariables(i).getValue());
         }
         if (getin().isMultipart())
         {
            getout().getContentType().setPrimaryType( "text"  );
            getout().getContentType().setSubType("plain");
            //getout().getContentType().setMultipartBoundary("xi-li-
hu-tu-yi-xian-tian-kai-zi-bian-zi-zao-boundary");
            int partscount = getin().countMultiParts();
            int outmlptindex = 0;
            for( int j = 0;j < partscount;j++)
            {
               if (getin().getMultiParts(j).getDisposition() != null)
               {
                  getout().getMultiParts(outmlptindex).setDisposition
(getin().getMultiParts(j).getDisposition());
               }
```

```
                else
                {
                }
                if (getin().getMultiParts(j).getDescription() != null)
                {
                   getout().getMultiParts(outmlptindex).setDescription
(     getin().getMultiParts(j).getDescription()     );
                }
                else
                {
                }
                if (getin().getMultiParts(j).getContentTypeString().in
dexOf("text") >= 0)
                {
                   getout().getMultiParts(outmlptindex).getContentType
().setPrimaryType(getin().getMultiParts(j).getContentType().getPrimar
yType() );
                   getout().getMultiParts(outmlptindex).getContentType
().setSubType(     getin().getMultiParts(j).getContentType().getSubTy
pe()  );
                   if (getin().getMultiParts(j).getContentType().getCo
ntentTransferEncoding() == null)
                   {
                      getout().getMultiParts(outmlptindex).getContentT
ype().setContentTransferEncoding("base64");
                   }
                   else
                   {
                   }
                   if (getin().getMultiParts(j).getContentType().getCh
arSet() != null &&
                   getin().getMultiParts(j).getContentType().getCharSe
t().compareToIgnoreCase("EUC_JP") == 0)
                   {
                      getout().getMultiParts(outmlptindex).getContentT
ype().setCharSet("SJIS");
                   }
                   else
                   {
                      if (getin().getMultiParts(j).getContentType().ge
tCharSet() != null)
                      {
                         getout().getMultiParts(outmlptindex).getConte
ntType().setCharSet(         getin().getMultiParts(j).getContentType(
).getCharSet()            );
                      }
                      else
                      {
                      }
                   }
                   getout().getMultiParts(outmlptindex).setDisposition
( getin().getMultiParts(j).getDisposition() );
                   getout().getMultiParts(outmlptindex).setDescription
("add a sample description in collab");
                   getout().getMultiParts(outmlptindex).setFileName(
       getin().getMultiParts(j).getFileName()  );
                   getout().getMultiParts(outmlptindex).setMimeBodyPar
tTextStringPayLoad(getin().getMultiParts(j).getMimeBodyPartTextString
PayLoad());
                   outmlptindex++;
                }
                else
                {
```

```
                    if (getin().getMultiParts(j).countNestedMultiPart()
 == 0)
                    {
                       if (getin().getMultiParts(j).getMimeBodyPartDeco
dedPayLoad() != null)
                       {
                          byte[] temppayload = getin().getMultiParts(ou
tmlptindex).getMimeBodyPartDecodedPayLoad();
                          getout().getMultiParts(outmlptindex).setMimeB
odyPartRawPayLoad(temppayload);
                          getout().getMultiParts(outmlptindex).setDispo
sition(   getin().getMultiParts(j).getDisposition()    );
                          getout().getMultiParts(outmlptindex).setDescr
iption(    getin().getMultiParts(j).getDescription()  );
                          getout().getMultiParts(outmlptindex).setFileN
ame(         getin().getMultiParts(j).getFileName() );
                          getout().getMultiParts(outmlptindex).getConte
ntType().setPrimaryType(     getin().getMultiParts(j).getContentType(
).getPrimaryType());
                          getout().getMultiParts(outmlptindex).getConte
ntType().setSubType(       getin().getMultiParts(j).getContentType().
getSubType()          );
                       }
                       else
                       {
                          byte[] temppayload = getin().getMultiParts(j)
.getMimeBodyPartRawPayLoad();
                          getout().getMultiParts(outmlptindex).setMimeB
odyPartRawPayLoad(temppayload);
                          getout().getMultiParts(outmlptindex).setDispo
sition(   getin().getMultiParts(j).getDisposition()    );
                          getout().getMultiParts(outmlptindex).setDescr
iption(    getin().getMultiParts(j).getDescription()  );
                          getout().getMultiParts(outmlptindex).setFileN
ame(         getin().getMultiParts(j).getFileName() );
                          getout().getMultiParts(outmlptindex).getConte
ntType().setPrimaryType(     getin().getMultiParts(j).getContentType(
).getPrimaryType());
                          getout().getMultiParts(outmlptindex).getConte
ntType().setSubType(       getin().getMultiParts(j).getContentType().
getSubType()          );
                       }
                       outmlptindex++;
                    }
                    else
                    {
                       int cntnestedpart = getin().getMultiParts(j).cou
ntNestedMultiPart();
                       for( int l = 0;l < cntnestedpart;l++)
                       {
                          if (getin().getMultiParts(j).getNestedMultiPa
rt(l).getMimeBodyPartDecodedPayLoad() != null)
                          {
                             byte[] temppayload = getin().getMultiParts
(j).getNestedMultiPart(l).getMimeBodyPartDecodedPayLoad();
                             getout().getMultiParts(outmlptindex).setMi
meBodyPartRawPayLoad(temppayload);
                             getout().getMultiParts(outmlptindex).setDi
sposition(   getin().getMultiParts(j).getNestedMultiPart(l).getDispos
ition()    );
                             getout().getMultiParts(outmlptindex).setDe
scription(    getin().getMultiParts(j).getNestedMultiPart(l).getDescr
iption()  );
```

```
                              getout().getMultiParts(outmlptindex).setFi
leName(         getin().getMultiParts(j).getNestedMultiPart(l).getFil
eName() );
                              getout().getMultiParts(outmlptindex).getCo
ntentType().setPrimaryType(     getin().getMultiParts(j).getNestedMul
tiPart(l).getContentType().getPrimaryType());
                              getout().getMultiParts(outmlptindex).getCo
ntentType().setSubType(       getin().getMultiParts(j).getNestedMulti
Part(l).getContentType().getSubType()          );
                           }
                           else
                           {
                              byte[] temppayload = getin().getMultiParts
(j).getNestedMultiPart(l).getMimeBodyPartRawPayLoad();
                              getout().getMultiParts(outmlptindex).setMi
meBodyPartRawPayLoad(temppayload);
                              getout().getMultiParts(outmlptindex).setDi
sposition(   getin().getMultiParts(j).getNestedMultiPart(l).getDispos
ition()    );
                              getout().getMultiParts(outmlptindex).setDe
scription(    getin().getMultiParts(j).getNestedMultiPart(l).getDescr
iption()  );
                              getout().getMultiParts(outmlptindex).setFi
leName(         getin().getMultiParts(j).getNestedMultiPart(l).getFil
eName() );
                              getout().getMultiParts(outmlptindex).getCo
ntentType().setPrimaryType(     getin().getMultiParts(j).getNestedMul
tiPart(l).getContentType().getPrimaryType());
                              getout().getMultiParts(outmlptindex).getCo
ntentType().setSubType(       getin().getMultiParts(j).getNestedMulti
Part(l).getContentType().getSubType()          );
                           }
                           outmlptindex++;
                        }
                     }
                  }
               }
            }
            else
            {
               System.err.println("Not Multiupart");
            }
            String strsinglebody;
            byte[] bytesinglebody;
            if (getin().isSingleBody())
            {
               getout().getContentType().setPrimaryType(getin().getConte
ntType().getPrimaryType());
               getout().getContentType().setSubType(getin().getContentTy
pe().getSubType());
               if (getin().getBody().getTextStringPayLoad() != null)
               {
                  strsinglebody=getin().getBody().getTextStringPayLoad()
;
                  getout().getBody().setTextStringPayLoad(strsinglebody)
;
                  if (getin().getContentType().getCharSet() != null &&
                  getin().getContentType().getCharSet() .compareToIgnore
Case("EUC_JP")==0)
                  {
                     getout().getContentType().setCharSet("SJIS");
                  }
                  else
                  {
```

```
                  if (getin().getContentType().getCharSet() != null)
                  {
                     getout().getContentType().setCharSet(getin().get
ContentType().getCharSet());
                  }
                  else
                  {
                  }
               }
            }
            else
            {
               if (getin().getBody().getTransferCodePayLoad() != null
)
               {
                  bytesinglebody = getin().getBody().getTransferCodeP
ayLoad();
                  getout().getBody().setRawPayLoad(bytesinglebody);
               }
               else
               {
                  bytesinglebody = getin().getBody().getRawPayLoad();
                  getout().getBody().setRawPayLoad(bytesinglebody);
               }
               System.err.println("Single body does not have text pay
load");
            }
         }
         else
         {
            System.err.println("is not single body type");
         }
         StringBuffer strbuf4 = new StringBuffer();
         if (getin().isUrlencoded())
         {
            for( int k = 0;k  < getin().countURLNameValueQueryPairs()
;k++)
            {
               strbuf4.append(" url name: "+getin().getURLNameValueQu
eryPairs(k).getName()+" value: "+getin().getURLNameValueQueryPairs(k)
.getValue());
            }
            getout().getContentType().setPrimaryType("text");
            getout().getContentType().setSubType("plain");
```

```
                getout().getBody().setTextStringPayLoad(strbuf4.toString(
));
            }
            else
            {
                System.err.println("not url encoded request");
            }
            getout().setJMSReplyTo(getin().getJMSReplyTo());
            getout().send(getout().getJMSReplyTo());
            ;
            return retBoolean;
        }

        public void userInitialize()
        {
        }

        public void userTerminate()
        {
        }

    }
```

Once the sample functions to your satisfaction, you can modify the schema to add functionality or create a new schema.

4.1.5 Test HTML Files

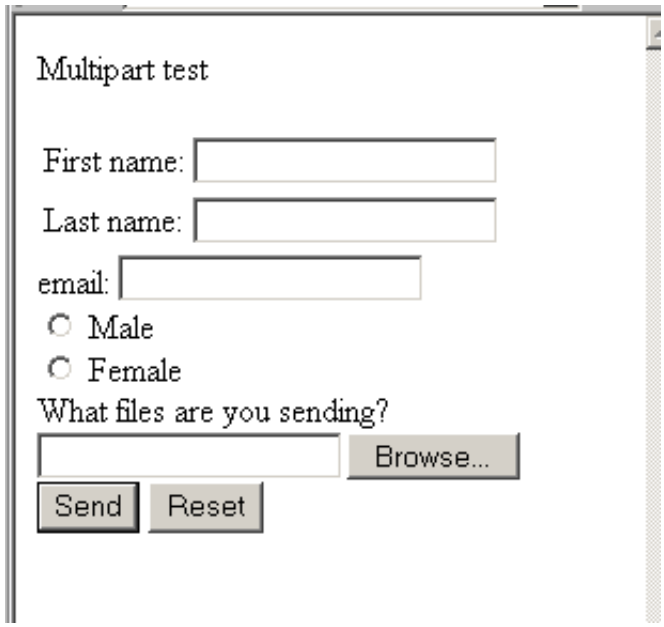
There are two test HTML files provided with the sample:

- testmulptfrm.html
- testurlencoded.html

testmulptfrm.html

The testmulptfrm.html file is provided for multi-part form data exchange and appears below:

Figure 15 testmultptfrm.html



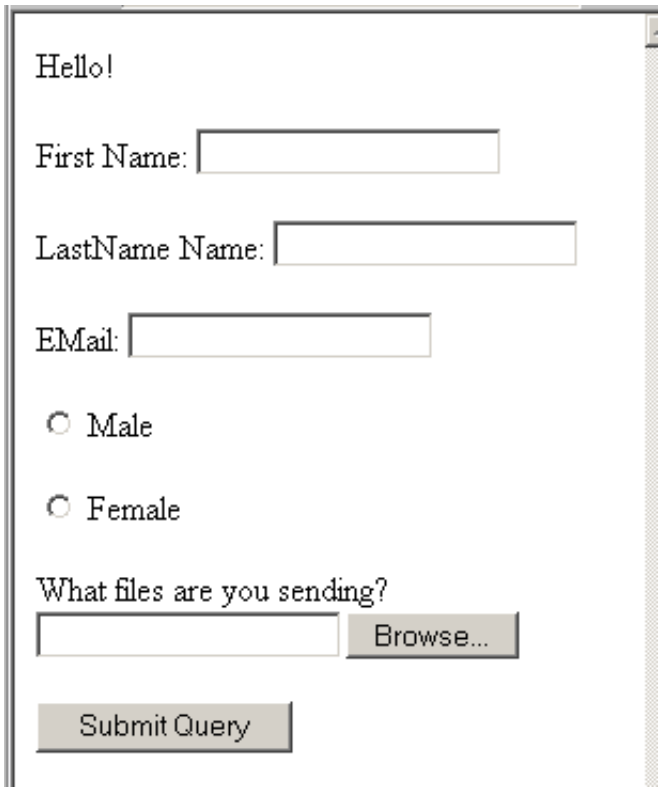
The HTML code follows:

```
<HTML>

<FORM ACTION="msisapiext/stc_iisapi.dll" METHOD="POST"
ENCTYPE="multipart/form-data">
Multipart test
<P>
<TABLE>
  <TR>
    <TD><LABEL for="fname">First name: </LABEL>
    <TD>  <INPUT type="text" name="firstname" id ="fname">
  <TR>
    <TD><LABEL for="lname">Last name: </LABEL>
    <TD><INPUT type="text" name="lastname" id="lname">
</TABLE>
    <LABEL for="email">email: </LABEL>
              <INPUT type="text" name="email"><BR>
    <INPUT type="radio" name="sex" value="Male"> Male<BR>
    <INPUT type="radio" name="sex" value="Female"> Female<BR>
    <LABEL for="filename">What files are you sending? </LABEL>
            <INPUT type="file" name="filename"><BR>
    <INPUT type="submit" value="Send"> <INPUT type="reset">
    </P>
</FORM>

</HTML>
```

testurlencoded.html

The testurlencoded.html file is provided for name/value pair form data exchange and appears below:

Figure 16 testurlencoded.html



The HTML code follows:

```
<FORM ACTION="msisapiext/stc_iisapi.dll" METHOD=POST>
Hello!
<P>
First Name:
<INPUT NAME=fname><BR>
<P>
LastName Name:
<INPUT NAME=lname><BR>
<P>
EMail:
<INPUT NAME=email><BR>
<p>
<INPUT type="radio" name="sex" value="Male"> Male<BR>
<p>
<INPUT type="radio" name="sex" value="Female"> Female<BR>
<P>
<LABEL for="filename">What files are you sending? </LABEL>
             <INPUT type="file" name="filename"><BR>
<P>
<INPUT TYPE=submit>
</FORM>
```

4.1.6 Message Routing to Multiple Collaborations

Only one type of topic can be published to each msisapiext directory. If the requirement is to publish many topics at the same time, the solution is to set up multiple msisapext directories. For example, msisapiext, msisapiext2, and so on. In each directory, there must be a copy of **stc_msapi.dll**, **stc_iisapi.dll**, **stc_msclient.dll**, **stc_mscommon.dll**, and **stc_iisapi.properties** (For more information see **"Configuring the Web Server Components" on page 18** .) Modify the **stc_iisapi.properties** file to point to the correct host and topic/queue to be published.

For example, if the user wants to publish two topics: Topic:etwebRequestETDTopic and Topic:webRequest, the user modifies the **stc_iisapi.properties** file, located in msisapiext directory, to specify Topic:etwebRequestETDtopic. Topic:webRequest is specified in the **stc_iisapi.properties** file, located in msisapiext2. Both msisapiext and msisapiext2 directories must contain the same version of the above mentioned files. The user can then create two Collaboration Rules to subscribe to etwebRequestETDTopic and webRequest.

To submit Topic:etwebRequestETDTopic via an HTTP client, the URL format used is:

```
hostname:port\msisapiext\stc_iisapi.dll
```

To submit Topic:webRequest via an HTTP client, the URL used is:

```
hostname:port\msisapiext2\stc_iisapi.dll
```

To achieve the above behavior, each web server must be configured to allow multiple msisapiext directories.

For IIS, create two virtual directories for msisapiext and msisapiext2.

Chapter 5

e*Way Java Classes and Methods

This chapter explains the Java classes and methods used by the e*Way Intelligent Adapter for the Microsoft Internet Information Server.

5.1 Classes and Methods: Overview

The e*Gate Enterprise Manager's Collaboration Editor graphical user interface (GUI) allows you to call Java methods by dragging and dropping an Event Type Definition (ETD) node into the **Rules** dialog box.

Note: *The node name can be different from the Java method name.*

After you drag and drop, the actual conversion takes place in the **.xsc** file. To view the **.xsc** file, use the Enterprise Manager's ETD Editor.

For example, if the node name is **encoding**, the associated **javaName** is **Encoding**. If you want to get the node value, use the Java method called **getEncoding**. If you want to set the node value, use the Java method called **setEncoding**.

Class Hierarchy

The MS IIS e*Way components support the following classes:

```
class java.lang.Object
   class com.stc.eways.webETD.Body
   class com.stc.eways.webETD.NameValue
   class com.stc.eways.webETD.OneMimeBodyPart
   class com.stc.eways.webETD.OneMimeBodyPart.NestedMultiPart
   class com.stc.jcsre.SimpleETDImpl (implements com.stc.jcsre.ETD)
      class com.stc.jcsre.MsgETDImpl
           class com.stc.eways.webETD.webReplyETD (implements
             com.stc.jcsre.ETD)
           class com.stc.eways.webETD.webRequestETD (implements
             com.stc.jcsre.ETD)
      class com.stc.eways.webETD.webETDContentType
      class com.stc.eways.webETD.webETDInternetHeaders
      class com.stc.eways.webETD.webReplyETD.ReplyMultiParts
      class com.stc.eways.webETD.webRequestETD.EnvironmentVariables
      class com.stc.eways.webETD.webRequestETD.MultiParts
      class com.stc.eways.webETD.webRequestETD.URLNameValueQueryPairs
```

5.2 Java Method List

This section lists the MS IIS e*Way Java methods by class.

5.2.1 Class Body

```
com.stc.eways.webETD
    java.lang.Object
com.stc.eways.webETD.Body
public class Body
extends java.lang.Object
```

Body

Description

Constructor.

Syntax

```
public Body()
```

getRawPayLoad

Description

Gets the raw payload byte array

Syntax

```
public byte[] getRawPayLoad()
```

Parameters

None.

Returns

byte array
Returns a byte array of the raw payload.

setRawPayLoad

Description

Sets the raw payload to the byte array

Syntax

```
public void setRawPayLoad(byte[] inrawpayload)
                          throws java.io.IOException
```

Parameters

Name	Type	Description
inrawpayload	byte[]	The raw payload to be set.

Returns

void

getTransferCodePayLoad

Description

Returns the decoded byte array (Request) or encoded byte array (Reply).

Syntax

```
public byte[] getTransferCodePayLoad()
```

Parameters

None.

Returns

byte array
Returns the decoded or encoded byte array.

setTnasferCodePayLoad

Description

Sets the encoded payload to the byte array.

Syntax

```
public void setTnasferCodePayLoad(byte[] inpayload)
                                  throws java.io.IOException
```

Parameters

Name	Type	Description
inpayload	byte[]	The payload to be set.

Returns

void

getTextStringPayLoad

Description

Gets the charSet decoded string payload.

Syntax

```
public java.lang.String getTextStringPayLoad()
```

Parameters

None.

Returns

java.lang.String
Returns the decoded string payload.

setTextStringPayLoad

Description

Sets the charSet encoded string payload.

Syntax

```
public void setTextStringPayLoad(java.lang.String inpayload)
```

Parameters

Name	Type	Description
inpayload	java.lang.String	The payload to be set.

Returns

void

marshal

Description

The outbound Collaboration controller will call this marshal method, you only need to either set rawpayload or text string payload.

Syntax

```
public byte[] marshal(java.lang.String encodingstyle,
   java.lang.String charset, boolean isText)
   throws com.stc.jcsre.MarshalException
```

Parameters

Name	Type	Description
encodingstyle	java.lang.String	The encoding style.
charset	java.lang.String	The charset.
isText	boolean	true indicates the payload is set to text.

Returns

byte array
Returns a byte array of flattened data.

---

unmarshal

Description

Takes a byte array and parses it into raw payload, decoded payload or string payload based on the content type.

Syntax

```
public void unmarshal(byte[] bytearraybodyin,webETDContentType
contenttype)
   throws com.stc.jcsre.UnmarshalException
```

Parameters

Name	Type	Description
bytearraybodyin	byte[]	An inbound byte array.
contenttype	weETDContentType	The content type to which to parse the data.

Returns

void

5.2.2 Class NameValue

```
java.lang.Object
    com.stc.eways.webETD.NameValue
public class NameValue
extends java.lang.Object
```

NameValue

Description

Constructor

Syntax

```
public NameValue()
```

NameValue

Description

Constructor

Syntax

```
public NameValue(java.lang.String nmin, byte[] valin)
```

Parameters

Name	Type	Description
nmin	java.lang.String	The name
valin	byte[]	A byte array of value

getName

Description

Gets the name of the variable.

Syntax

```
public java.lang.String getName()
```

Parameters

None.

Return Values

java.lang.String
Returns the name of the variable.

getValue

Description

Gets the value of this pair.

Syntax

```
public byte[] getValue()
```

Parameters

None.

Return Values

byte array
Returns value as a byte array.

setIsURLencoded

Description

Marks this pair as URL encoded.

Syntax

```
public void setIsURLencoded(boolean isEncoded)
```

Parameters

Name	Type	Description
isEncoded	boolean	true indicates that the part is URL encoded.

Returns

void

---

marshal

Description

Marshal serializes this object into a byte array.

Syntax

```
public byte[] marshal()
   throws com.stc.jcsre.MarshalException
```

Parameters

None.

Returns

byte array
Returns the corresponding deserialized NameValue object.

---

unmarshal

Description

Unmarshal deserializes the byte array passed into this object.

Syntax

```
public void unmarshal(byte[] inputEvent)
   throws com.stc.jcsre.UnmarshalException
```

Parameters

Name	Type	Description
inputEvent	byte[]	A byte array deserialized into a NameValue object..

5.2.3 Class OneMimeBodyPart

```
com.stc.eways.webETD
java.lang.Object
   com.stc.eways.webETD.OneMimeBodyPart
public class OneMimeBodyPart
extends java.lang.Object
```

OneMimeBodyPart

Description

Constructor.

Syntax

```
public OneMimeBodyPart()
```

getMimeBodyPart

Description

Returns a javax.mail.internet.MimeBodyPart object.

Syntax

```
public com.stc.eways.webETD.MimeBodyPart getMimeBodyPart()
```

Parameters

None.

getMimeBodyPartRawPayLoad

Description

Returns the raw payload byte array.

Syntax

```
public byte[] getMimeBodyPartRawPayLoad()
   throws javax.mail.MessagingException,
          java.io.IOException
```

Parameters

None.

Return

byte[]
Returns a byte array of the raw payload.

setMimeBodyPartRawPayLoad

Description

Sets the raw payload.

Syntax

```
public void setMimeBodyPartRawPayLoad(byte[] inbytearray)
   throws javax.mail.MessagingException,
          java.io.IOException
```

Parameters

Name	Type	Description
inbytearray	byte[]	A byte array of the raw payload.

Parameters:

---

getMimeBodyPartDecodedPayLoad

Description

Returns the decoded payload byte array.

Syntax

```
public byte[] getMimeBodyPartDecodedPayLoad()
   throws javax.mail.MessagingException,
          java.io.IOException
```

Parameters

None.

Return

byte array
Returns a byte array of the decoded payload.

---

getMimeBodyPartTextStringPayLoad

Description

Returns the string payload.

Syntax

```
public java.lang.String getMimeBodyPartTextStringPayLoad()
   throws javax.mail.MessagingException,
          java.io.IOException
```

Parameters

None.

Return

java.lang.String
Returns the string payload.

---

setMimeBodyPartTextStringPayLoad

Description

Sets the string payload.

Syntax

```
public void setMimeBodyPartTextStringPayLoad(java.lang.String instr)
   throws javax.mail.MessagingException,
          java.io.IOException
```

Parameters

Name	Type	Description
instr	java.lang.String	The inbound string to set.

Return

void

isMimeBodyPart

Description

Returns true if the part is a mime part, or false if it is URL encoded.

Syntax

```
public boolean isMimeBodyPart()
```

Parameters

None.

Returns

boolean
Returns true if the part is a mime part, otherwise, returns false.

isMimeType

Description

Queries whether this part of the specified MIME type. This method compares only the primaryType and subType.

Syntax

```
public boolean isMimeType(java.lang.String typein)
   throws javax.mail.MessagingException
```

Parameters

Name	Type	Description
typein	java.lang.String	The part to query.

Returns

boolean
Returns true if the part is a MIME type.

getDisposition

Description

Returns the content disposition parameter as a string.

Syntax

```
public java.lang.String getDisposition()
   throws javax.mail.MessagingException
```

Parameters

None.

Return

java.lang.String
Returns the content disposition parameter as a string.

setDisposition

Description

Sets the disposition of the content type parameter.

Syntax

```
public void setDisposition(java.lang.String instr)
   throws javax.mail.MessagingException
```

Parameters

Name	Type	Description
instr	java.lang.String	The content type to be set.

Return

void

setContentID

Description

Sets the ContentID content type parameter.

Syntax

```
public void setContentID(java.lang.String instr)
   throws javax.mail.MessagingException
```

Parameters

Name	Type	Description
instr	java.lang.String	The contentID to be set.

Return

void

getContentID

Description

Gets the contentID content type parameter.

Syntax

```
public java.lang.String getContentID()
   throws javax.mail.MessagingException
```

Parameters

None.

Return

java.lang.String
Returns the contentID content type parameter.

getContentTypeString

Description

Returns the contentType string.

Syntax

```
public java.lang.String getContentTypeString()
   throws javax.mail.MessagingException
```

Parameters

None.

Return

java.lang.String
Returns the contentType string.

setContentMD5

Description

Sets the contentMD5 for this body part.

Syntax

```
public void setContentMD5(java.lang.String instr)
   throws javax.mail.MessagingException
```

Parameters

Name	Type	Description
instr	java.lang.String	The contentMD5 to be set.

Return

void

getContentMD5

Description

Gets the contentMD5 for this body part.

Syntax

```
public java.lang.String getContentMD5()
   throws javax.mail.MessagingException
```

Parameters

None.

Return

java.lang.String
Returns the contentMD5 for this body part.

getDescription

Description

Gets the description of this body part.

Syntax

```
public java.lang.String getDescription()
   throws javax.mail.MessagingException
```

Parameters

None.

Return

java.lang.String
Returns the description for the body part.

setDescription

Description

Sets the description of this body part.

Syntax

```
public void setDescription(java.lang.String instr)
   throws javax.mail.MessagingException
```

Parameters

Name	Type	Description
instr	java.lang.String	The description to be set.

Return

void

getEncoding

Description

Gets the encoding of this body part.

Syntax

```
public java.lang.String getEncoding()
   throws javax.mail.MessagingException
```

Parameters

None.

Return

java.lang.String
Returns the encoding for the body part.

getFileName

Description

Gets the filename parameter for this body part.

Syntax

```
public java.lang.String getFileName()
   throws javax.mail.MessagingException
```

Parameters

None.

Return

java.lang.String
Returns the filename parameter for the body part.

setFileName

Description

Sets the filename of this body part.

Syntax

```
public void setFileName(java.lang.String instr)
   throws javax.mail.MessagingException
```

Parameters

Name	Type	Description
instr	java.lang.String	The filename to be set.

Return

void

getContentType

Description

Gets the content type as a webETDContentType object for this body part.

Syntax

```
public webETDContentType getContentType()
   throws javax.mail.MessagingException
```

Parameters

None.

Return

webETDContentType
Returns the content type as a webETDContentType object for the body part.

countNestedMultiPart

Description

Counts the number of parts in a nested multipart.

Syntax

```
public int countNestedMultiPart()
```

Parameters

None.

Return

int
Returns the number of parts in the nested multipart form, if none, returns zero.

getNestedMultiPart

Description

Returns the nested multipart.

Syntax

```
public OneMimeBodyPart getNestedMultiPart(int index)
```

Parameters

Name	Type	Description
index	int	The index number for the nested multipart to be returned.

Return

OneMimeBodyPart
Returns the nested multipart object.

marshal

Description

Serializes this object into a byte array.

Syntax

```
public byte[] marshal(boolean buildmbponly)
   throws com.stc.jcsre.MarshalException,
          javax.mail.MessagingException,
          java.io.IOException
```

Parameters

Name	Type	Description
buildmbponly	boolean	The object to be deserialized. The build MimeBodyPart does not return a byte array.

Return

byte[]
Returns a byte array corresponding to a deserialized OneMimeBodyPart object, else null.

unmarshal

Description

Deserializes the byte array passed into this object.

Syntax

```
public void unmarshal(byte[] abodyins)
   throws javax.mail.MessagingException,
          java.io.IOException
```

Parameters

Name	Type	Description
abodyins	byte[]	The byte array to be deserialized into a OneMimeBodyPart object.

5.2.4 Class OneMimeBodyPart.NestedMultiPart

```
com.stc.eways.webETD
Class OneMimeBodyPart.NestedMultiPart
java.lang.Object
   com.stc.eways.webETD.OneMimeBodyPart.NestedMultiPart
Enclosing class:OneMimeBodyPart
public class OneMimeBodyPart.NestedMultiPart
extends java.lang.Object
```

OneMimeBodyPart.NestedMultiPart

Description

Constructor.

Syntax

```
public OneMimeBodyPart.NestedMultiPart()
```

---

setCurrIndex

Description

Sets the current index point.

Syntax

```
public void setCurrIndex(int i)
```

Parameters

Name	Type	Description
i	int	The index number for the nested multpart to be set.

Return

void

getSize

Description

Gets the size.

Syntax

```
public int getSize()
```

Parameters

None.

Return

int
Returns the size of the object.

getPart

Description

Gets the part.

Syntax

```
public OneMimeBodyPart getPart()
```

Parameters

None.

Return

OneMimeBodyPart
Returns the OneMimeBodyPart object.

unmarshal

Description

Deserializes the byte array passed into this object.

Syntax

```
public void unmarshal(byte[] bytearraybodyin)
   throws com.stc.jcsre.UnmarshalException,
          java.io.IOException,
          javax.mail.MessagingException
```

Parameters

Name	Type	Description
bytearraybodyin	byte[]	The byte array to be deserialized into a OneMimeBodyPart object.

5.2.5 Class webReplyETD

```
java.lang.Object
   com.stc.jcsre.StimpleETDImpl
       com.stc.jcsre.MsgETDImpl
           com.stc.eways.webETD.webReplyETD
All Implemented Interfaces: com.stc.jcsre.ETD,
com.stc.jcsre.ETDConstansts
public class webReplyETD
extends com.stc.jcsre.MsgETDImpl
implements com.stc.jcsre.ETD
```

---

webReplyETD

Description

Constructor.

Syntax

```
public webReplyETD()
```

---

getJMSReplyTo

Description

Gets the JMSReplyTo field. This is normally obtained from the corresponding webRequestETD.

Syntax

```
public java.lang.String getJMSReplyTo()
```

Parameters

None.

Return Values

java.lang.String
Returns the JMSReplyTo field.

---

setJMSReplyTo

Description

Sets the JMSReplyTo filed. This is normally obtained from the corresponding webRequestETD.

Syntax

```
public void setJMSReplyTo(java.lang.String replyTo)
```

Parameters

Name	Type	Description
replyTo	java.lang.String	The JMSReplyTo field to set.

Return Values

void

send

Description

Overrides the super's send( ) method to work with JMS. You must call setJMSReplyTo before call this method.

Syntax

```
public void send()
```

Parameters

None.

Return

void

Specified by

`send` in interface `com.stc.jcsre.ETD`

Overrides

`send` in class `com.stc.jcsre.MsgETDImpl`

getBody

Description

Gets the body of this reply, if the reply is of a discrete type (text/plain, image/jpeg).

Syntax

```
public Body getBody()
```

Parameters

None.

Return

Returns the body of this reply if the reply is discrete type. For example, text/plain or image/jpeg.

getContentType

Description

Gets the content type of this reply, as a webETDContentType object.

Syntax

```
public webETDContentType getContentType()
```

Parameters

None.

Return Values

webETDContentType
Returns the webETDContentType object.

addHeader

Description

Adds a name/value pair as a new header to this reply.

Syntax

```
public void addHeader (java.lang.String name, java.lang.String value)
```

Parameters

Name	Type	Description
name	java.lang.String	The name string
value	java.lang.String	The value string

Return Values

void

setHeader

Description

Sets the name/value pair header. If the header name already exists, the associated value is overwritten.

Syntax

```
public void setHeader(java.lang.String name, java.lang.String value)
```

Parameters

Name	Type	Description
name	java.lang.String	The name of the header to set.
value	java.lang.String	The value to set for the header.

Return Values

void

getHeader

Description

Gets the first header line by the name and delimiter.

Syntax

```
public java.lang.String getHeader(java.lang.String name,
java.lang.String delim)
```

Parameters

Name	Type	Description
name	java.lang.String	The name of the header to get.
delim	java.lang.String	The delimiter.

Return Values

java.lang.String
Returns the first header line value.

getMultiParts

Description

Gets the value for the indexed part, provided it is multipart data.

Syntax

```
public OneMimeBodyPart getMultiParts(int index)
```

Parameters

Name	Type	Description
index	int	The index point for the part.

Return Values

OneMimeBodyPart
Returns the value for the part.

isMultipart

Description

Queries whether the request, originated as multipart data.

Syntax

```
public boolean isMultipart()
```

Parameters

None.

Return Values

boolean
Returns true if the request originated as multipart data.

countMultiParts

Description

Counts the number of parts contained in the multipart body.

Syntax

```
public int countMultiParts()
```

Parameters

None.

Return Values

int
Returns the number of parts.

marshal

Description

Serializes the object into a byte array.

Syntax

```
public byte[] marshal()
   throws com.stc.jcsre.MarshalException
```

Parameters

None.

Return Values

byte array
Returns a byte array of the serialized object.

Specified by

```
marshal in interface com.stc.jcsre.ETD
```

Overrides

`marshal` in class `com.stc.jcsre.SimpleETDImpl`

5.2.6 Class webRequestETD

```
java.lang.Object
   com.stc.jcsre.SimpleETDImpl
       com.stc.jcsre.MsgETDImpl
           com.stc.eways.webETD.webRequestETD
All Implemented Interfaces:
com.stc.jcsre.ETD, com.stc.jcsre.ETDConstants
public class webRequestETD
extends com.stc.jcsre.MsgETDImmpl
implements com.stc.jcsre.ETD
```

---

webRequestETD

Description

Constructor.

Syntax

```
public webRequestETD()
```

---

getJMSReplyTo

Description

Gets the JMSReplyTo for this Event.

Syntax

```
public java.lang.String getJMSReplyTo()
```

Parameters

None.

Return Values

java.lang.String
Returns JMSReplyTo string.

---

getContentType

Description

Gets the ContentType object.

Syntax

```
public webETDContentType getContentType()
```

Parameters

None.

Return Values

webETDContentType
Returns webETDContentType object.

getRequestMethod

Description

Gets request method string.

Syntax

```
public java.lang.String getRequestMethod()
```

Parameters

None.

Returns

java.lang.String
Returns the request method string.

getEnvironmentVariables

Description

Gets the environment variables object.

Syntax

```
public webRequestETD.EnvironmentVariables getEnvironmentVariables(int
index)
```

Parameters

Name	Type	Description
index	int	The index point.

Return Values

webRequestETD.EnvirnmentVariables
Returns the webRequestETD.EnvironmentVariable object.

countEnvironmentVariables

Description

Counts the number of environment variables that are part of the SeeBeyond Java Messaging Service (JMS) property passed in. Each environment variable is a name value pair. For example, SERVER_PROTOCOL=HTTP/1.1.

Syntax

```
public int countEnvironmentVariables()
```

Parameters

None.

Return Values

int
Returns the number of environment variables.

getURLNameValueQueryPairs

Description

Gets the specified name/value pair, URL decoded.

Syntax

```
public webRequestETD.URLNameValueQueryPairs
getURLNameValueQueryPairs(int index)
```

Parameters

Name	Type	Description
index	int	The index point.

Return Values

webRequestETD.URLNameValueQueryPairs
Returns the specified name/value pair.

isSingleBody

Description

Queries whether the request originated as Application/x-www-urlencoded data.

Syntax

```
public boolean isSingleBody()
```

Parameters

None.

Return Values

boolean
Returns true if the original request was sent in Application/x-www-urlencoded format.

getBody

Description

Gets the body of this webRequestETD, provided the content type is discrete (text/plain or image/jpeg).

Syntax

```
public Body getBody()
```

Parameters

None.

Return Values

Body
Returns the body of the webRequestETD.

isUrlencoded

Description

Queries whether the HTTP request (GET) is URL encoded.

Syntax

```
public boolean isUrlencoded()
```

Parameters

None.

Return Value

boolean
Returns true to indicate that the HTTP request (GET) is URL encoded.

countURLNameValueQueryPairs

Description

Counts the number of environment variables.

Syntax

```
public int countURLNameValueQueryPairs()
```

Parameters

None.

Return Value

int
Returns the number of environment variables.

getMultiParts

Description

Gets the specified part.

Syntax

```
public OneMimeBodyPart getMultiParts(int index)
```

Parameters

Name	Type	Description
index	int	The index point.

Return Values

OneMimeBodyPart
Returns the OneMimeBodyPart object.

isMultipart

Description

Queries whether the request originated as Multipart/form-data.

Syntax

```
public boolean isMultipart()
```

Parameters

None.

Return Values

boolean
Returns true if the request originated as Multipart/form-data.

countMultiParts

Description

Counts the number of MIME parts.

Syntax

```
public int countMultiParts()
```

Parameters

None.

Return Values

int
Returns the number of MIME parts.

unmarshal

Description

Deserializes the byte array passed into this object.

Syntax

```
public void unmarshal(byte[] inputEvent)
   throws com.stc.jcsre.UnmarshalException
```

Parameters

Name	Type	Description
inputEvent	byte array	The input Event to be deserialized.

Return Values

void

Specified by

`unmarshal` in interface `com.stc.jcsre.ETD`

Overrides

`unmarshal` in class `com.stc.jcsre.SimpleETDImpl`

5.2.7 Class webETDContentType

```
com.stc.eways.webETD
java.lang.Object
   com.stc.eways.webETD.webETDContentType
public class webETDContentType
extends java.lang.Object
```

webETDContentType

Description

Constructor.

Syntax

```
public webETDContentType()
```

unmarshal

Description

Deserializes an object input put as a string.

Syntax

```
public void unmarshal(java.lang.String instr)
   throws javax.mail.internet.ParseException
```

Parameters

Name	Type	Description
instr	java.lang.String	The input as a string.

Return Values

void

toString

Description

Returns a string representation of the object.

Syntax

```
public java.lang.String toString()
```

Parameters

None.

Return Values

java.lang.String
Returns a string representation of the object.

Overrides

`toString` in class `java.lang.Object`

getPrimaryType

Description

Gets the primary type in a media type, such as, text in text/plain.

Syntax

```
public java.lang.String getPrimaryType()
```

Parameters

None.

Returns

java.lang.String
Returns the primary type as a string.

setPrimaryType

Description

Sets the primary type in the object. For example, text in text/plain.

Syntax

```
public void setPrimaryType(java.lang.String instr)
```

Parameters

Name	Type	Description
instr	java.lang.String	The primary type to set.

Return Value

void

getSubType

Description

Gets the sub type in media type. For example, plain in text/plain.

Syntax

```
public java.lang.String getSubType()
```

Parameters

None.

Return Values

java.lang.String
Returns the sub type in a media type.

setSubType

Description

Sets the sub type in the object. For example, plain in text/plain.

Syntax

```
public void setSubType(java.lang.String instr)
```

Parameters

Name	Type	Description
instr	java.lang.String	The sub type to set.

Return Values

void

getCharSet

Description

Gets the charSet in a media type. For example, EUC_JP.

Syntax

```
public java.lang.String getCharSet()
```

Parameters

None.

Return Values

java.lang.String
Returns the charSet value.

setCharSet

Description

Sets the charSet in a media type. For example, EUC_JP.

Syntax

```
public void setCharSet(java.lang.String instr)
```

Parameters

Name	Type	Description
instr	java.lang.String	The charSet type to set.

Return Values

void

getMultipartBoundary

Description

Gets the boundary parameter for a content type.

Syntax

```
public java.lang.String getMultipartBoundary()
```

Parameters

None.

Return Values

java.lang.String
Returns the boundary parameter.

setMultipartBoundary

Description

Sets the boundary parameter in a content type.

Syntax

```
public void setMultipartBoundary(java.lang.String instr)
```

Parameters

Name	Type	Description
instr	java.lang.String	The boundary to set.

Return Values

void

getContentTransferEncoding

Description

Gets the content transfer encoding parameter for a content type. For example, base64.

Syntax

```
public java.lang.String getContentTransferEncoding()
```

Parameters

None.

Return Values

java.lang.String
Returns the content transfer encoding parameter.

setContentTransferEncoding

Description

Sets the content transfer encoding parameter in a content type. For example, base64.

Syntax

```
public void setContentTransferEncoding(java.lang.String instr)
```

Parameters

Name	Type	Description
instr	java.lang.String	The content-transfer-encoding to set.

Return Values

void

getContentTypeParameter

Description

Gets a specified parameter in a content type. For example, ContentID.

Syntax

```
public java.lang.String getContentTypeParameter(java.lang.String nm)
```

Parameters

Name	Type	Description
nm	java.lang.String	The name of the parameter to get.

Return Values

java.lang.String
Returns a parameter as a string.

setContentTypeParameter

Description

Sets the parameter value in a content type. For example, ContentID.

Syntax

```
public void setContentTypeParameter(java.lang.String nm,
                                    java.lang.String val)
```

Parameters

Name	Type	Description
nm	java.lang.String	The name of the parameter to set.
val	java.lang.String	The value to be set.

Return Values

void

---

marshal

Description

Serializes the object into a byte array.

Syntax

```
public byte[] marshal()
   throws com.stc.jcsre.MarshalException
```

Parameters

None.

Return Values

byte array
Returns a byte array of the serialized object.

5.2.8 Class webETDInternetHeaders

```
com.stc.eways.webETD
java.lang.Object
   com.stc.eways.webETD.webETDInternetHeaders
public class webETDInternetHeaders
extends java.lang.Object
```

---

webETDInternetHeaders

Description

Constructors.

Syntax

```
public webETDInternetHeaders()
```

---

addHeader

Description

Adds a name/value pair header. The delimiter will default.

Syntax

```
public void addHeader(java.lang.String nm,
                      java.lang.String val)
```

Parameters

Name	Type	Description
nm	java.lang.String	The name of the parameter to set.
val	java.lang.String	The value to be set.

Return Values

void

setHeader

Description

Sets a name/value pair header delimiter. If the header already exists, it is overwritten.

Syntax

```
public void setHeader(java.lang.String name,
                      java.lang.String value)
```

Parameters

Name	Type	Description
name	java.lang.String	The name of the parameter to set.
value	java.lang.String	The value to be set.

Return Values

void

getHeader

Description

Gets the value of a header and delimiter. If multiple header lines match, the first one is returned.

Syntax

```
public java.lang.String getHeader(java.lang.String nm,
                                  java.lang.String delim)
```

Parameters

Name	Type	Description
nm	java.lang.String	The name of the parameter to get.
delim	java.lang.String	The delimiter to get.

Return Values

java.lang.String
Returns the value of a header and delimiter. If multiple header lines match, the first one is returned.

---

getAsInternetHeaders

Description

Gets the headers as internet headers.

Syntax

```
public javax.mail.internet.InternetHeaders getAsInternetHeaders()
```

Parameters

None.

Return Values

javax.mail.internet
Returns the headers as javax.mail.internet.InternetHeaders.

---

marshal

Description

Serializes the object into a byte array.

Syntax

```
public byte[] marshal()
   throws com.stc.jcsre.MarshalException
```

Parameters

None.

Return Values

byte array
Returns a byte array of the serialized object.

5.2.9 Class webReplyETD.ReplyMultiParts

```
com.stc.eways.webETD
java.lang.Object
   com.stc.eways.webETD.webReplyETD.ReplyMultiParts
Enclosing class:
webReplyETD
   public class webReplyETD.ReplyMultiParts
extends java.lang.Object
```

webReplyETD.ReplyMultiParts

Description

Constructor.

Syntax

```
public webReplyETD.ReplyMultiParts()
```

setCurrIndex

Description

Sets the current index.

Syntax

```
public void setCurrIndex(int i)
```

Parameters

Name	Type	Description
i	int	The index point to set.

Return Values

void

getSize

Description

Gets the size of the part.

Syntax

```
public int getSize()
```

Parameters

None.

Return Values

int

Returns the size of the part.

getPart

Description

Gets the OneMimeBodyPart object.

Syntax

```
public OneMimeBodyPart getPart()
```

Parameters

None.

Return Values

OneMimeBodyPart
Returns the OneMimeBodyPart object.

---

marshal

Description

Serializes the object into a byte array, based on the content type.

Syntax

```
public byte[] marshal(webETDContentType topcontenttype)
   throws com.stc.jcsre.MarshalException
```

Parameters

Name	Type	Description
topcontenttype	webETDContentType	The webETDContentType object.

Return Values

byte array
Returns a byte array of the serialized object.

5.2.10 Class webRequestETD.EnvironmentVariables

```
com.stc.eways.webETD
java.lang.Object
   com.stc.eways.webETD.webRequestETD.EnvironmentVariables
Enclosing class:
   webRequestETD
public class webRequestETD.EnvironmentVariables
extends java.lang.Object
```

---

webRequestETD.EnvironmentVariables

Description

Constructor.

Syntax

```
public webRequestETD.EnvironmentVariables()
```

setCurrIndex

Description

Sets the current index point.

Syntax

```
public void setCurrIndex(int i)
```

Parameters

Name	Type	Description
i	int	The index point to set.

Return Values

void

addEnvVarPair

Description

Adds an Environmental variable pair.

Syntax

```
protected void addEnvVarPair(NameValue newpair)
```

Parameters

Name	Type	Description
newpair	NameValue	The name value object to be added.

Return Values

void

getSize

Description

Gets the size of the variable.

Syntax

```
public int getSize()
```

Parameters

None.

Return Values

int
Returns the size of the variable.

---

getName

Description

Gets the name of the variable.

Syntax

```
public java.lang.String getName()
```

Parameters

None.

Return Values

java.lang.String
Returns the name of this environment variable.

---

getValue

Description

Gets the value of the variable.

Syntax

```
public java.lang.String getValue()
```

Parameters

None.

Return Values

java.lang.String
Returns the value for the environment variable.

5.2.11 Class webRequestETD.MultiParts

```
com.stc.eways.webETD
java.lang.Object
   com.stc.eways.webETD.webRequestETD.MultiParts
Enclosing class:
webRequestETD
public class webRequestETD.MultiParts
extends java.lang.Object
```

---

webRequestETD.MultiParts

Description

Constructor.

Syntax

```
public webRequestETD.MultiParts()
```

setCurrIndex

Description

Sets the current index point.

Syntax

```
public void setCurrIndex(int i)
```

Parameters

Name	Type	Description
i	int	The index point to set.

Return Values

void

getSize

Description

Gets the size of the part.

Syntax

```
public int getSize()
```

Parameters

None.

Return Values

int

Returns the size of the part.

unmarshal

Description

Deserializes the byte array passed into this object.

Syntax

```
public void unmarshal(byte[] bytearraybodyin)
   throws com.stc.jcsre.UnmarshalException,
          java.io.IOException,
          javax.mail.MessagingException
```

Parameters

Name	Type	Description
bytearraybodyin	byte array	The object to be deserialized.

Return Values

void

---

getPart

Description

Gets the part.

Syntax

```
public OneMimeBodyPart getPart()
```

Parameters

None.

Return Values

OneMimeBodyPart
Returns the OneMimeBodyPart object.

5.2.12 Class webRequestETD.URLNameValueQueryPairs

```
com.stc.eways.webETD
java.lang.Object
   com.stc.eways.webETD.webRequestETD.URLNameValueQueryPairs
Enclosing class:
webRequestETD
public class webRequestETD.URLNameValueQueryPairs
extends java.lang.Object
```

---

webRequestETD.URLNameValueQueryPairs

Description

Constructor.

Syntax

```
public webRequestETD.URLNameValueQueryPairs()
```

---

getSize

Description

Gets the size of the name/value pair.

Syntax

```
public int getSize()
```

Parameters

None.

Return Values

int
Returns the size of the name/value pair.

---

getName

Description

Gets the name of the name/value pair.

Syntax

```
public java.lang.String getName()
```

Parameters

None.

Return Values

java.lang.String
Returns the name of the name/value pair.

---

getValue

Description

Gets the value for the name/value pair.

Syntax

```
public java.lang.String getValue()
```

Parameters

None.

Return Values

java.lang.String
Returns the value for the name/value pair.

---

setCurrIndex

Description

Sets the current index point.

Syntax

```
public void setCurrIndex(int i)
```

Parameters

Name	Type	Description
i	int	The index point to set.

Return Values

void

unmarshal

Description

Separates the URL encoded string into name/value pairs.

Syntax

```
public void unmarshal(java.lang.String urlencodedstring)
   throws com.stc.jcsre.UnmarshalException
```

Parameters

Name	Type	Description
urlencodedstring	java.lang.String	The string to be broken down into a name/value pair.

Return Values

void

Index

A

B

C

H

I

J

L

M

N

O

P

R

S

T

U

W