SeeBeyond™ eBusiness Integration Suite

e*Way Intelligent Adapter for axion4 User’s Guide

Release 4.5.2

SEEBEYOND™

Version 20020509135020.

Contents

Chapter 1

Introduction

This document describes how to install and configure the Java-enabled e*Way Intelligent Adapter for axion4 Gateway.

1.1 Overview

The Java-enabled e*Way Intelligent Adapter for Axion4 Gateway (axion4 gateway e*Way) acts as an interface to **axion4gstp's Axion4 Gateway** product. This provides connectivity to axion4gstp's Transaction Flow Manager (TFM) over a secure TCP/IP Virtual Private Network.

The axion4 gateway e*Way meets the following business requirements:

- Single pipe. The e*Way is a single pipe into the axion4 network for an organization, supporting multiple systems feeding into the gateway and dispersing messages and acknowledgments to multiple systems from the gateway.
- STP enabler. The STP (straight-through-processing) is defined as a state in which messages can be exchanged from one system to another without manual intervention, including error conditions. Error conditions may require manual intervention for correction, but manual intervention will not terminate the flow unless necessary.
- Configurable. The e*Way is configurable via the GUI. Complicated programming tasks are not required by you in order for the e*Way to perform standard/default behavior.

The e*Way utilizes a set of axion4gstp APIs to create a bi-directional link. The axion4 gateway e*Way provides a "base of functionality" that ensures reliable and efficient communication of axion4 messages that is configurable via the e*Way Properties window. A Java class exposes all of the methods used to create the standard functionality to enable users to create custom functionality.

The axion4 ETD library provides access to the TFM Message library.

1.1.1 Operational Overview

Global economic and financial trends show an increase in cross-border investing driven by public pension reform and by the increasing complexity of investment strategies.

The current cross-border trade processing environment is characterized by manual procedures, multiple service providers, incompatible databases, lack of standardization, excessive costs and a high rate of expensive failed trades.

The impact of each of these characteristics is exacerbated by global economic and financial trends, such as the increase in the complexity of investment strategies, public pension reform, and the aging of the world population.

The Global Straight Through Processing Association Limited (GSTPA) is an industry association incorporated in the United Kingdom to address these trends. The membership of the GSTPA consists of Investment Managers, Broker/Dealers, and Global Custodians involved in the processing of cross-borders trades.

GSTPA developed and patented the Transaction Flow Manager (TFM), which is the core of its GSTP solution is designed to assist Investment Managers, Broker/Dealers and Global Custodians in inter-operating with each other in a standard, automated way in support of their post-trade/pre-settlement activities.

The Transaction Flow Manager (TFM) is accessed through a secure TCP/IP Virtual Private Network and will track, number, timestamp and route cross-border trade information and will perform a number of matching, enrichment and other value-added functions for the benefit of participants.

The axion4 gateway is an axion4gstp provided TFM Access Module. A set of APIs and functions facilitate a gateway that allows participants to connect to the TFM. The axion4 gateway uses a store and forward mechanism to ensure reliability.

1.1.2 Functional Design

The axion4 gateway e*Way provides two messaging modes in the default configuration, **sending** and **receiving**. Only one messaging mode can be supported in the e*Way at a time. The Java class through which the e*Way exposes the axion4 gateway API enables you to create customized versions of these modes.

Each mode has three messaging states:

- initiate
- messaging
- terminate

Messaging States

Initiate State

The *initiate* state is where the actual connection is established to the gateway and TFM. Successful completion of this state will transition the e*Way into the *messaging* state. Failure causes the program to shut down.

During the collection of parameters, the e*Way will verify that all of the parameter values are valid. For example, a passed queue name exists in the current schema. The usual timeout/recovery/retry functionality will be applied.

Messaging State

The *messaging* state is the main state of this protocol for either flow direction. Once a connection is established, this state deals with the reliable and efficient transmission of messages to and from e*Gate and to the axion4 gateway.

The e*Way will fall back into the initiation state any time the connection is "dropped." And, if the connection is unsuccessful, after the usual timeout/recovery/retry functionality, the e*Way will send an alert and shut down.

Terminate State

The e*Way transitions to the *terminate* state when it receives a shutdown command from the control broker or when it gets into an unrecoverable error state. While in this state, the e*Way follows the necessary shutdown procedures and sends an alert if an error occurs.

Sending - Outbound Functionality

The outbound functionality begins with the initiate state. The e*Way reads in the appropriate parameters from the configuration file and creates or modifies the e*Way's functionality accordingly.

Note: *Multiple instances of the e*Way may be created and configured in a subscriber pool for higher through-put and reliability.*

The e*Way then tries to connect to the axion4 gateway and establish a connection to the TFM. If unsuccessful, the e*Way automatically retries up to three times to connect, at which point if the connection is not established, the e*Way publishes a gatewayErr to the error queue with the content of the error.

If connection is successful, the e*Way transitions to the messaging state. The e*Way checks the queue for any messages to be sent. If a message is available in the queue, the e*Way calls the send API configured for Autocommit.

Note: *It is the responsibility of the user to validate the TFM message before it is passed to the e*Way. The e*Way performs no validation upon the message, the message is passed as a valid TFM message.*

If the send is successful, the e*Way sends an application level acknowledgment back to a specified queue (if configured) and immediately checks the outbound queue for another message.

If the send is unsuccessful due to a communication problem, the e*Way either reverts to the initiate state to re-establish a connection or fails if the communication error is unrecoverable. If the error is non-communication related, based on the failed message delivery continuation parameter, the e*Way publishes the message with the result of the call prepended to the error queue. If a duplicate message is received, the error message contains the message with the result prepended along with the messagStatus API call result. If the failed message continuation parameter is reached, an alert is sent to the Control Broker to request the user manually shut down the e*Way.

Forwarding Application Level Acknowledgments

The e*Way contains a parameter in the outbound mode that directs the e*Way to send the result of the send API back into e*Gate, specifically a queue, to be forwarded to the originating system.

Failed Message Delivery Continuation

In the outbound direction, the e*Way can optionally be configured to either skip a message that cannot be delivered due to a non-communication problem or shut down. The message is skipped after it is safely stored in an error queue. The message contains the result of the API prepended to it before storage.

The e*Way provides a parameter that represents the number of messages that can be skipped before the e*Way shuts down. This parameter has the following values:

- 0 = no limit to the number of skipped messages.
- 1 = shut down the e*Way upon the first failed message.
- >1 =the actual number of skipped messages that will be tolerated.

If the e*Way receives a shut down command, an alert is sent and the standard shut down procedures are followed.

Receiving - Inbound Functionality

Whereas multiple instances were optional in the outbound e*Way, they are required in the inbound flow. The fact that the commit function does not have a context or handle to prevent all outstanding transactions in any thread from being committed (even those that are still in process) causes the need to create a reliable transfer mechanism in which the message obtained from the retrieve call is only committed after the message is safely stored in the e*Gate queue. To handle each message individually, multiple instances must be executed.

The inbound phases starts with the initiate state. The e*Way reads in the appropriate parameters from the e*Way configuration file and then creates or modifies the e*Way's functionality accordingly.

The e*Way then tries to connect to the axion4 gateway to establish a connection to the TFM. If unsuccessful, the e*Way automatically retries up to three times, at which time the e*Way publishes a gatewayErr to the error queue with the content of the error.

If successful, the e*Way transitions to the messaging state, where it checks the gateway for the specified message category at a specified interval. The e*Way calls the API configured to retrieve only one message at a time.

If no message is returned the e*Way waits for a specified sleep time-out and tries again. If a message is returned, it is written to the inbound queue. If successful, the e*Way commits the retrieve, otherwise, publishes a gatewayErr to the error queue with the content of the error.

If a message is retrieved, the e*Way calls the retrieve function until it does not return any additional messages.

1.1.3 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 expert-level knowledge of the Java Programming Language; to have expert-level knowledge of Windows and UNIX operations and administration; to be thoroughly familiar with axion4 gateway API protocol and to be thoroughly familiar with Windows-style GUI operations.

1.1.4 Components

The Java-enabled axion4 e*Way is comprised of the following components:

- **axion4 e*Way Java class**. This component wraps the *axion4 gateway* API and exposes its methods as Java methods as well as methods that are necessary for reliable and efficient communication with the *axion4 gateway*.
- **Inbound Collaboration**. This collaboration is an implementation of the axion4 Java class which is used to conform to the inbound functionality.
- **Outbound Collaboration**. This collaboration is an implementation of the axion4 Java class which is used to conform to the outbound functionality.
- **axion4 Java ETD Library**. This library was created from DTDs contained in the TFM Message Manuals.
- **axion4 DTD Library**: This library was created from TFM Message Manuals.
- **axion4 e*Way Connection**: The e*Way Connection is used to communicate with the gateway/TFM.

A complete list of installed files appears in **Table 1 on page 14**.

1.2 System Requirements

To use the Java-enabled version of the axion4 gateway e*Way, you need the following:

- An e*Gate Participating Host, version 4.5.1 or later running on Solaris 2.6
- 100 MB free disk space, plus the following

  for e*Way executable, configuration, library, and script files. The above amounts of disk space will be required on both the Participating and the Registry Host.

Note: *Additional disk space will be required to process and queue the data that this e*Way processes; the amount necessary will vary based on the type and size of the data being processed, and any external applications performing the processing.*

- A TCP/IP network connection.
- axion4ETD Library (can be installed simultaneously).
- The e*Way must be run in the environment configured with Swiftnet profile, (filename/password) otherwise the API calls will fail.

The client components of the databases with which the e*Way interfaces have their own requirements; see that system's documentation for more details.

1.2.1 Supported Operating Systems

The axion4 e*Way is currently supported on the following operating systems:

- Solaris 2.6

1.2.2 External System Requirements

The external system requirements are different for a GUI host machine—specifically a machine running the ETD Editor and the Java Collaboration Editor GUIs—versus a participating host which is used solely to run the e*Gate schema.

- axion4 gateway and shared libraries
- Operational axion4 network connection
- Java Run-time environment (JRE) 1.3
- Java JDK 1.3. The JDK can be installed during the e*Gate GUI installation process if it hasn't been installed already.
- The LD_LIBRARY_PATH must be set to point to the Axion4 Gateway library files. For example,

```
/opt/Swift/axion4:opt/SUNWspro/lib:/opt/axion4gstp/a4g/lib
```

Chapter 2

Installation

This chapter describes how to install the axion4 e*Way.

Note: *If installing the e*Way on a 4.5.0 Participating Host, before installing, back-up stcutil.jar and stcexception.jar. After installing the e*Way, replace the newly installed .jar files with the previously backed up files. The .jar files are located:*

```
eGate\Server\Registry\Repository\default\classes
```

2.1 Installing the axion4 gateway e*Way on UNIX

2.1.1 Pre-installation

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

2.1.2 Installation Procedure

To install the axion4 gateway e*Way on a UNIX system

1 Log in on the workstation containing the CD-ROM drive, and insert the CD-ROM into the drive.

2 If necessary, mount the CD-ROM drive.

3 At the shell prompt, type

cd /cdrom

4 Start the installation script by typing

setup.sh

5 A menu of options will appear. Select the "install e*Way" option. Then follow any additional on-screen directions.

Note: *Be sure to install the e*Way files in the suggested "client" installation directory. The installation utility detects and suggests the appropriate installation directory.*

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

2.2 Files/Directories Created by the Installation

The axion4 gateway e*Way installation process will install the following files within the e*Gate directory tree. Files will be installed within the “egate\client” tree on the Participating Host and committed to the “default” schema on the Registry Host.

Table 1 Files created by the installation of the Java-enabled axion4 e*Way

e*Gate Directory	Files(s)
classes\	stcaxion4.jar
collaboration_rules\axion4	Axion4InboundCollaboration.class Axion4InboundCollaborationBase.class Axion4InboundCollaboration.ctl Axion4OutboundCollaboration.class Axion4OutboundCollaborationBase.class Axion4OutboundCollaboration.ctl
configs\axion4	axion4.def
bin\	stc_axion4jni.so

Table 1 Files created by the installation of the Java-enabled axion4 e*Way

e*Gate Directory	Files(s)
etd\axion4	AccountingInformation.jar AccountInformation.xsc AdviceThirdParty_FX_Deal.jar AdviceThirdParty_FX_Deal.xsc AlertMsgTimeExpiry.jar AlertMsgTimeExpiry.xsc Allocations.jar Allocation.xsc AllocationToGC.jar AllocationToGC.xsc AllocNotification.jar AllocNotification.xsc Axion4OutputXML.xsc Axion4Generic.jar Axion4Generic.xsc BD_MultiPart.jar BD_MultiPart.xsc BON.jar BON.xsc DeadlineChangeNotification.jar DeadlineChangeNotification.xsc Error.jar Error.xsc IM_MultiPart.jar IM_MultiPart.xsc NetProceeds.jar NetProceeds.xsc NetProceedsMatchStatus.jar NetProceedsMatchStatus.xsc NetProceedsStatusAdvice.jar NetProceedsStatusAdvice NOE.jar NOE.xsc Notif_ThirdParty_FX_Deal.jar Notif_ThirdParty_FX_Deal.xsc NotificationAllegedBON.jar NotificationAllegedBON.xsc NotificationAllegedNOE.jar NotificationAllegedNOE.xsc NotificationNetProceeds.jar NotificationNetProceeds.xsc NotificationPreAllocation.jar NotificationPreAllocation.xsc NotificationRejectRevokeReject.jar NotificationRejectRevokeReject.xsc NotificationSettlementDetails.jar NotificationSettlementDetails.xsc Pre_Allocation.jar Pre_Allocation.xsc RejectRevokeReject.jar

Table 1 Files created by the installation of the Java-enabled axion4 e*Way

e*Gate Directory	Files(s)
etd\axion4 (cont'd)	RejectRevokeReject.xsc SettlementDetails.jar SettlementDetails.xsc Success.xsc TradeMatchStatusAdvice.jar TradeMatchStatusAdvice.xsc Axion4OutputXML.jar SettlementDetailsStatusAdvice.jar SettlementDetailsStatusAdvice.xsc SettlementReleaseDetails.jar SettlementReleaseDetails.xsc StatusChange.jar StatusChange.xsc Success.jar
dtd\axion4	AccountingInformation.dtd AdviceThirdParty_FX_Deal.dtd AlertMsgTimeExpiry.dtd Allocations.dtd AllocationToGC.dtd AllocNotification.dtd BD_MultiPart.dtd BON.dtd DeadlineChangeNotification.dtd Error.dtd IM_MultiPart.dtd NetProceeds.dtd NetProceedsMatchStatus.dtd NetProceedsStatusAdvice.dtd NOE.dtd Notif_ThirdParty_FX_Deal.dtd NotificationAllegedBON.dtd NotificationAllegedNOE.dtd NotificationNetProceeds.dtd NotificationPreAllocation.dtd NotificationRejectRevokeReject.dtd NotificationSettlementDetails.dtd Pre_Allocation.dtd RejectRevokeReject.dtd SettlementDetails.dtd SettlementDetailsStatusAdvice.dtd SettlementReleaseDetails.dtd StatusChange.dtd Success.dtd TradeMatchStatusAdvice.dtd

Table 1 Files created by the installation of the Java-enabled axion4 e*Way

e*Gate Directory	Files(s)
monk_scripts\common	Axion4OutputXML.ssc Axion4Generic.ssc AccountingInformation.ssc AdviceThirdParty_FX_Deal.ssc Allocations.ssc AllocationToGC.ssc AllocNotification.ssc BD_MultiPart.ssc BON.ssc DeadlineChangeNotification.ssc Error.ssc IM_MultiPart.ssc NetProceeds.ssc NetProceedsMatchStatus.ssc NetProceedsStatusAdvice.ssc NOE.ssc Notif_ThirdParty_FX_Deal.ssc NotificationAllegedBON.ssc NotificationAllegedNOE.ssc NotificationNetProceeds.ssc NotificationPreAllocation.ssc NotificationRejectRevokeReject.ssc NotificationSettlementDetails.ssc Pre_Allocation.ssc RejectRevokeReject.ssc SettlementDetails.ssc SettlementDetailsStatusAdvice.ssc SettlementReleaseDetails.ssc StatusChange.ssc Success.ssc TradeMatchStatusAdvice.ssc

Chapter 3

e*Way Setup

This chapter summarizes the initial setup procedures to create both the axion4 e*Way and the file-based e*Ways.

3.1 Overview

After installing the Axion4 e*Way, you must perform an initial setup for it to work with your system. A wide range of setup options allow the e*Way to conform to your system's operational characteristics and your facility's operating procedures.

The topics discussed in this chapter include the following:

3.2 Setting Up the e*Way

3.2.1 Defining e*Way Components

The first step in implementing an e*Way is to define the e*Way component using the e*Gate Enterprise Manager.

To create an e*Way

1. Select the e*Gate Enterprise Manager Navigator's **Components** tab.
2. Open the host on which you want to create the e*Way.
3. Select the Control Broker that will manage the new e*Way.
4. On the Palette, click **Create a New e*Way**.
5. Enter the name of the new e*Way, then click **OK**.

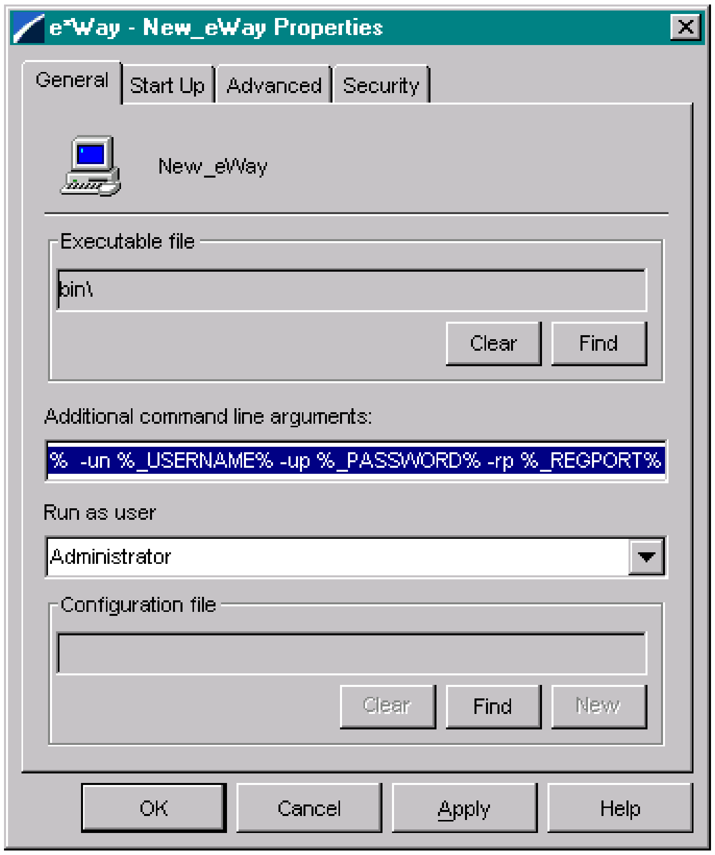
3.2.2 Modifying e*Way Properties

Note: *Selecting the executable file should be the first configuration procedure you perform once you have created the e*Way.*

To modify any e*Way properties

1. Select the Navigator's **Components** tab.
2. Open the host on which the desired e*Way runs.
3. Open the Control Broker that manages the e*Way.
4. Select the desired e*Way.
5. Right-click on the e*Way and select **Properties** to edit the e*Way's properties. The properties dialog will open to the **General** tab (shown in Figure 1).

Figure 1 e*Way Properties (General Tab)



bin\

% -un %_USERNAME% -up %_PASSWORD% -rp %_REGPORT%

Administrator

6 Make the desired modifications, then click **OK**.

Note: *When you shut down an e*Way and open its property sheet in the e*Gate Enterprise Manager, once you click* **OK** *or* **Apply***, the e*Way immediately restarts. This action only happens if the e*Way is in autostart mode. After you click* **OK** *or* **Apply***, the Registry is automatically updated with any changes, if you made them using the e*Way Editor.*

3.2.3 Selecting an Executable File

Selecting the executable file is the first and most important step in configuring the e*Way. This step determines what type of e*Way will run and thus what type of external system or communications protocol it will support.

You must know which executable file to select before you perform this procedure.

To select an e*Way's executable file

1 Display the e*Way's properties (see the **procedure on page 19**).

2 On the General tab, under **Executable File**, click **Find**.

3 Use the file selection dialog box to select the executable files. All e*Way executable files have a **.exe** extension.

Note: *You must use the* **Find** *button to select the executable file. You cannot type its name directly into the Executable File box. The Axion4 e*Way uses the executable* ***stceway.exe*** *and is referred to as a Multi-Mode e*Way. The File-based e*Ways use the executable* ***stcewfile.exe****.*

3.2.4 Selecting or Creating a Configuration File

After you have selected an executable file, you must select or create a configuration file that will contain the operating parameters for the e*Way.

To select an existing configuration file

1. Display the e*Way's properties (see the **procedure on page 19**).
2. On the **General** tab, under **Configuration File**, click **Find**.
3. Use the file selection dialog box to select the desired file (***.cfg**).
4. **Exit** the e*Way Editor.

Note: *You must use the* **Find** *button to select the configuration file. You cannot type its name directly into the Configuration File box.*

To create a configuration file

1. Display the e*Way's properties (see the **procedure on page 19**).
2. On the **General** tab, under **Configuration File**, click **Find**.
3. Use the file selection dialog box to select a default configuration (template) file.

Note: *All e*Way default configuration files have a* **.def** *extension, and are intended to be used as templates.*

4. Use the e*Way Editor to change the default configuration parameters as required .
5. Edit the **Additional Command Line Arguments** box to include any arguments you require (see the following procedure).
6. **Save** the file with a **.cfg** extension and **exit** the e*Way Editor.

Note: *You can use the* **FIND** *or* **NEW** *button to select the configuration file. You cannot type its name directly into the Configuration File box.*

3.2.5 Changing Command-line Parameters

Most SeeBeyond e*Ways require only the default command-line parameters shipped with the e*Gate Enterprise Manager. Use the procedure in this section only if the e*Way you are configuring requires special command-line options, or if you are directed to do so by SeeBeyond support personnel.

To change an e*Way's command-line options

1. Display the e*Way's properties (see the **procedure on page 19**).

2 On the **General** tab, edit the **Additional Command Line Arguments** box to include the arguments you require. Unless you have a specific need to do so, do not change any of the existing parameters; add any new parameters to the end of the command line.

3.2.6 Changing the User Name

Like all e*Gate executable components, e*Ways run under an e*Gate user name. By default, all e*Ways run under the **Administrator** user name. You can change this if your site's security procedures so require.

To change the "run as" user name

1 Display the e*Way's properties (see the **procedure on page 19**).

2 On the **General** tab, use the Run As list to select the e*Gate user under whose name this component will run.

See the *e*Gate Integrator System Administration and Operations Guide* for more information on the e*Gate security system.

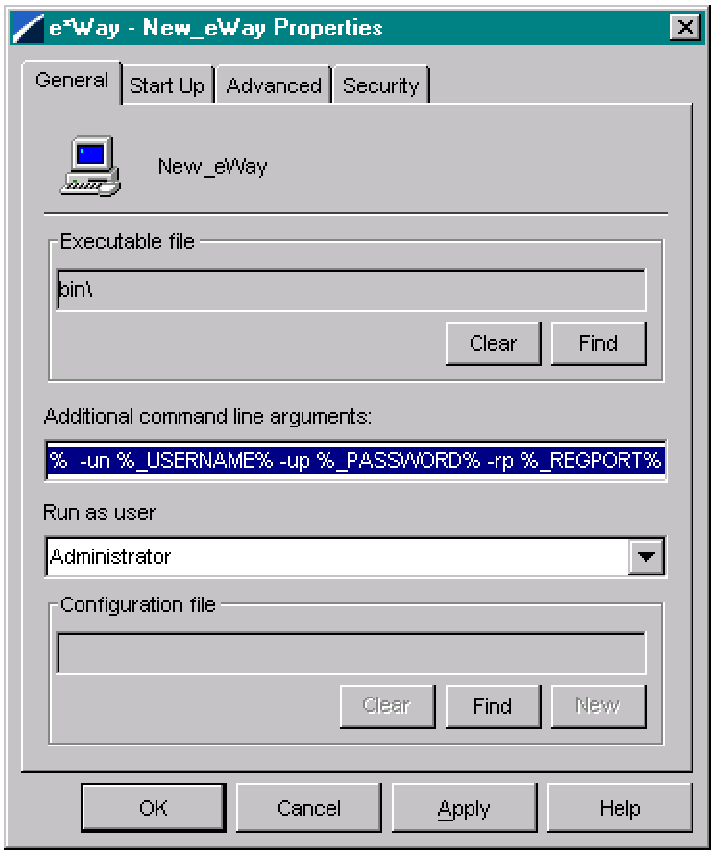
3.2.7 Setting Startup Options or Schedules

SeeBeyond e*Ways can be started or stopped by any of the following methods:

- The Control Broker can start the e*Way automatically whenever the Control Broker starts.
- The Control Broker can start the e*Way automatically whenever it detects that the e*Way terminated execution abnormally.
- The Control Broker can start or stop the e*Way on a schedule that you specify.
- Users can start or stop the e*Way manually using an interactive monitor.

You determine how the Control Broker starts or shuts down an e*Way using options on the e*Way properties **Start Up** tab (see Figure 2). See the *e*Gate Integrator System Administration and Operations Guide* for more information about how interactive monitors can start or shut down components.

Figure 2 e*Way Properties (Start-Up Tab)



e*Way - New_eWay Properties

General | Start Up | Advanced | Security

New_eWay

Executable file: bin\

Additional command line arguments: % -un %_USERNAME% -up %_PASSWORD% -rp %_REGPORT%

Run as user: Administrator

Configuration file:

To determine whether the e*Way starts automatically when the Control Broker starts

1 Display the e*Way's properties (see the **procedure on page 19**).

2 Select the **Start Up** tab.

3 To activate this feature, check **Start automatically**.

4 To deactivate this feature, clear the **Start automatically** check box.

5 Click **OK**.

To determine whether the e*Way will be restarted automatically

1 Display the e*Way's properties (see the **procedure on page 19**).

2 Select the **Start Up** tab.

3 To activate this feature, check **Restart after abnormal termination** and set the desired number of retries and retry interval.

4 To deactivate this feature, clear the **Restart** check box.

5 Click **OK**.

Note: *The "auto restart" feature will not automatically restart the e*Way if the e*Way is shut down manually by an interactive monitor.*

*If the e*Way is shut down and you make any configuration changes using the e*Gate Enterprise Manager, the Control Broker will automatically restart the e*Way when the configuration changes are recorded in the e*Gate Registry. If you do not want the e*Way to restart when configuration changes are made, disable this feature before configuring the e*Way.*

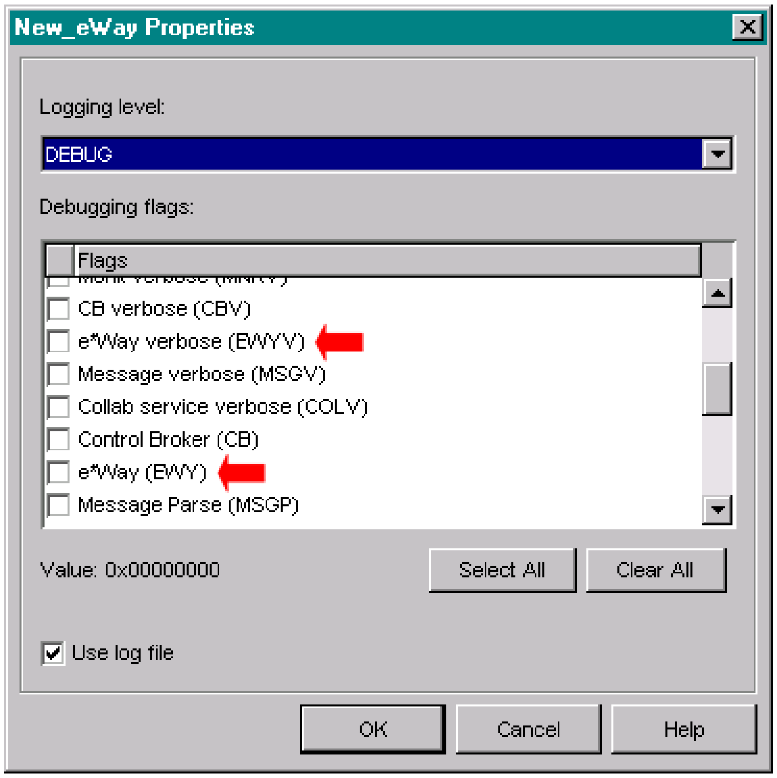
3.2.8 Activating or Modifying Logging Options

Logging options enable you to troubleshoot problems with the e*Way and other e*Gate components.

To set the e*Way debug level and flag

1. Select the Navigator's **Components** tab.
2. Open the host on which the desired e*Way runs.
3. Open the Control Broker that manages the e*Way.
4. Select the desired e*Way.
5. On the Toolbar, click **Properties** to edit the e*Way's properties.
6. Select the **Advanced** tab.
7. Click **Log**. The dialog window will appear as in Figure 3.
8. Select **DEBUG** for the **Logging level**.
9. Select either **e*Way (EWY)** or **e*Way Verbose (EWYV)** for the **Debugging flag.** Note that the latter will have a significant impact on system performance.
10. Click **OK**.

Figure 3 e*Way Properties (Advanced Tab - Log Option)



New_eWay Properties

Logging level: DEBUG

Debugging flags:

Flags
- [ ] CB verbose (CBV)
- [ ] e*Way verbose (EWYV)
- [ ] Message verbose (MSGV)
- [ ] Collab service verbose (COLV)
- [ ] Control Broker (CB)
- [ ] e*Way (EWY)
- [ ] Message Parse (MSGP)

Value: 0x00000000

- [x] Use log file

The other options apply to other e*Gate components and are activated in the same manner. See the *e*Gate Integrator Alert and Log File Reference* for additional information concerning log files, logging options, logging levels, and debug flags.

3.2.9 Activating or Modifying Monitoring Thresholds

Monitoring thresholds enable you to monitor the throughput of the e*Way. When the monitoring thresholds are exceeded, the e*Way will send a Monitoring Event to the Control Broker, which will be routed to the e*Gate Monitor or any of a number of destinations.

1. Display the e*Way's properties (see the **procedure on page 19**).
2. Select the **Advanced** tab.
3. Click **Thresholds**.
4. Select the desired threshold options and click **OK**.

See the *e*Gate Integrator Alert and Log File Reference* for more information concerning threshold monitoring, routing specific notifications to specific recipients, or for general information about e*Gate's monitoring and notification system.

3.3 Troubleshooting the e*Way

Because of the flexibility provided for customization of SeeBeyond e*Ways, it is impossible to provide a comprehensive guide to troubleshooting. However, this section provides guidelines to follow when troubleshooting of the e*Way's operation or performance. In the initial stages of developing your e*Gate Integrator system administration system, most problems with e*Ways can be traced to configuration.

3.3.1 In the Enterprise Manager

- Does the e*Way have the correct Collaborations assigned?
- Do those Collaborations use the correct Collaboration Services?
- Is the logic correct within any Collaboration Rules script employed by this e*Way's Collaborations?
- Do those Collaborations subscribe to and publish Events appropriately?
- Are all the components that "feed" this e*Way properly configured, and are they sending the appropriate Events correctly?
- Are all the components that this e*Way "feeds" properly configured, and are they subscribing to the appropriate Events correctly?

3.3.2 In the e*Way Editor

- Check that all configuration options are set appropriately.
- Check that all settings you changed are set correctly.
- Check all required changes to ensure they have not been overlooked.
- Check the defaults to ensure they are acceptable for your installation.

3.3.3 On the e*Way's Participating Host

- Check that the Participating Host is operating properly, and that it has sufficient disk space to hold the IQ data that this e*Way's Collaborations publish.

3.3.4 In the e*Way's External Application

- Check that the application is configured correctly, is operating properly, and is sending or receiving the correct data appropriately.
- Check that the connection between the external application and the e*Way is functioning appropriately.
- Once the e*Way is up and running properly, operational problems can be due to:
  - External influences (network or other connectivity problems).
  - Problems in the operating environment (low disk space or system errors)
  - Problems or changes in the data the e*Way is processing.
  - Corrections required to Collaboration Rules scripts that become evident in the course of normal operations.

One of the most important tools in the troubleshooter's arsenal is the e*Way log file. See the *e*Gate Integrator Alert and Log File Reference Guide* for an extensive explanation of log files, debugging options, and using the e*Gate monitoring system to monitor operations and performance.

Chapter 4

Multi-Mode e*Way Configuration

This chapter describes how to configure the Multi-Mode e*Way, which works with the axion4 e*Way Connection to communicate with the external systems.

4.1 Configuring the Multi-Mode e*Way

e*Way properties are set using the Enterprise Manager.

To create and configure a New e*Way:

1. Select the Navigator's Components tab.
2. Open the host on which you want to create the e*Way.
3. On the Palette, click the icon to create a new **e*Way**.
4. Type the name of the new e*Way, and then click **OK**.
5. Select the new component, then click to edit its properties.
6. When the e*Way Properties window opens, click the **Find** button beneath the **Executable File** field, and select an executable file. (**stceway.exe** is located in the "bin\" directory.)
7. Under the **Configuration File** field, click the **New** button. When the Settings page opens, set the configuration parameters for this configuration file.
8. After selecting the desired parameters, save the configuration file. Close the **.cfg** file and select **OK** to close the e*Way Properties Window.

The e*Way configuration parameters are organized into the following section:

- JVM Settings

4.1.1 JVM Settings

The JVM Settings control basic Java Virtual Machine settings.

JNI DLL Absolute Pathname

Description

Specifies the absolute pathname to where the JNI DLL installed by the *Java 2 SDK* is located on the Participating Host. This parameter is **mandatory**.

Required Values

A valid pathname.

Additional Information

The JNI dll name varies on different O/S platforms:

OS	Java 2 JNI DLL Name
Solaris 2.6	libjvm.so

The value assigned can contain a reference to an environment variable, by enclosing the variable name within a pair of percent (%) symbols. For example:

```
%MY_JNIDLL%
```

Such variables can be used when multiple Participating Hosts are used on different platforms.

To ensure that the JNI DLL loads successfully, the Dynamic Load Library search path environment variable must be set appropriately to include all the directories under the Java 2 SDK (or JDK) installation directory that contain shared libraries (UNIX) or DLLs (NT).

CLASSPATH Prepend

Description

Specifies the paths to be prepended to the CLASSPATH environment variable for the Java VM.

Required Values

An absolute path or an environmental variable. This parameter is optional.

Additional Information

If left unset, no paths will be prepended to the CLASSPATH environment variable.

Existing environment variables may be referenced in this parameter by enclosing the variable name within a pair of percent (%) symbols. For example:

```
%MY_PRECLASSPATH%
```

CLASSPATH Override

Description

Specifies the complete CLASSPATH variable to be used by the Java VM. This parameter is optional. If left unset, an appropriate CLASSPATH environment variable (consisting of required e*Gate components concatenated with the system version of CLASSPATH) will be set.

Note: *All necessary JAR and ZIP files needed by both e*Gate and the Java VM must be included. It is advised that the **CLASSPATH Prepend** parameter should be used.*

Required Values

An absolute path or an environmental variable. This parameter is optional.

Additional Information

Existing environment variables may be referenced in this parameter by enclosing the variable name within a pair of (%) symbols. For example:

```
%MY_CLASSPATH%
```

CLASSPATH Append From Environment Variable

Description

Specifies the whether or not to append the classpath from the environment variable.

Required Values

YES or No. The default is NO.

Initial Heap Size

Description

Specifies the value for the initial heap size in bytes. If set to 0 (zero), the preferred value for the initial heap size of the Java VM will be used.

Required Values

An integer between 0 and 2147483647. This parameter is optional.

Maximum Heap Size

Description

Specifies the value of the maximum heap size in bytes. If set to 0 (zero), the preferred value for the maximum heap size of the Java VM will be used.

Required Values

An integer between 0 and 2147483647. This parameter is optional.

Maximum Stack Size for Native Threads

Description

Specifies the value of the maximum stack size in bytes for native threads. If set to 0 (zero), the default value will be used.

Required Values

An integer between 0 and 2147483647. This parameter is optional.

Maximum Stack Size for JVM Threads

Description

Specifies the value of the maximum stack size in bytes for JVM threads. If set to 0 (zero), the preferred value for the maximum heap size of the Java VM will be used.

Required Values

An integer between 0 and 2147483647. This parameter is optional.

Class Garbage Collection

Description

Specifies whether the Class Garbage Collection will be done automatically by the Java VM. The selection affects performance issues.

Required Values

YES or **NO**.

Garbage Collection Activity Reporting

Description

Specifies whether garbage collection activity will be reported for debugging purposes.

Required Values

YES or **NO**.

Asynchronous Garbage Collection

Description

Specifies whether asynchronous garbage collection activity will be reported for debugging purposes.

Required Values

YES or **NO**.

Report JVM Info and all Class Loads

Description

Specifies whether the JVM information and all class loads will be reported for debugging purposes.

Required Values

YES or **NO**.

Disable JIT

Description

Specifies whether the Just-In-Time (JIT) compiler will be disabled.

Required Values

YES or **NO**.

Note: *This parameter is not supported for Java Release 1.*

Remote Debugging Port Number

Description

Specifies the port number for remote debugging of the JVM.

Required Values

An integer between 2000 and 65536.

Suspend Option For Debugging

Description

Specifies whether or not to suspend the JVM for debugging.

Required Values

YES or NO.

Chapter 5

e*Way Connection Configuration

This chapter describes how to configure the axion4 Gateway e*Way Connections.

5.1 e*Way Connection

e*Way Connections are set using the Enterprise Manager.

To create and configure e*Way Connections

1. In the Enterprise Manager's **Component** editor, select the **e*Way Connection** folder.
2. On the palette, click on the icon to create a new **e*Way Connection**.
3. The **New e*Way Connection Component** dialog box opens. Type a name for the **e*Way Connection** , and then Click **OK**.
4. Double-click on the new **e*Way Connection**.
5. The **e*Way Connection Properties** dialog box opens.
6. From the **e*Way Connection Type** drop-down box, select **axion4**.
7. Enter the **Event Type "get"** interval in the dialog box provided.
8. From the **e*Way Connection Configuration File**, click **New** to create a new Configuration File for this Connection. (To use an existing file, click **Find**.)

The e*Way Connections configuration parameters are organized into the following sections:

- connector
- Axion4 Configure
- Axion4 Send
- Axion4 Retrieve
- Axion4 Terminate

5.1.1 connector

This section contains a set of top level parameters:

- type

- class
- Property.Tag

type

Description

Specifies the type of connection.

Required Values

axion4. The value defaults to axion4.

class

Description

Specifies the class name of the axion4 Client connector object.

Required Values

A valid package name. The default is **com.stc.eways.axion4.Axion4Connector**.

Property.Tag

Description

Specifies the data source identity.

Required Values

A valid data source package name.

5.1.2 Axion4 Configure

This section contains a set of top level parameters used by axion4:

- File Name
- Password
- Number of Trade Processes
- Number of Query Processes
- Number of Report Processes
- Maximum Number of Gateway Configure Retries

file name

Description

Specifies the fully qualified path of the SNL security profile file.

Required Values

A valid path containing the profile file.

password

Description

Specifies the password necessary to access the SNL security profile file specified above.

Required Values

A string. A valid SNL password.

Number of Trade Processes

Description

Specifies the number of outbound trade processes to be started when the configure method is called.

Required Values

An integer between 1 and 864000. Entering “ ” (blank) will result in the use of the value specified in the axion4 gateway profile.

Number of Query Processes

Description

Specifies the number of outbound query processes to be started when the configure method is called.

Required Values

An integer between 1 and 864000. Entering “ ” (blank) will result in the use of the value specified in the axion4 gateway profile.

Number of Report Processes

Description

Specifies the number of outbound report processes to be started when the configure method is called.

Required Values

An integer between 0 and 864000. Entering “ ” (blank) will result in the use of the value specified in the axion4 gateway profile.

Maximum Number of Gateway Configure Retries

Description

Specifies the maximum number of gateway configure retries if the gateway is down. Once the maximum number of gateway configure retries is reached, an alert is sent to the user and the gateway goes into a blocked state until the user can shut it down.

Required Values

An integer between 0 and 864000. Entering “ ” (blank) will result in the use of the value specified in the axion4 gateway profile.

5.1.3 Axion4 Send

The parameters in this section specify the information required for the connection to access the external systems to send messages to the axion4 gateway:

- Priority
- Forward Send API Acknowledgment
- Failed Message Continuation

Priority

Description

Sets the default priority of messages sent to axion4.

Required Values

Normal or **High**.

st invoke the 'commit' method to effect the changes in the database.

Forward Send API Acknowledgement

Description

Specifies whether to publish the result of the send API.

Required Values

Yes or **No**.

Failed Message Continuation

Description

Specifies the number of messages that can be skipped before the e*Way shuts down.

Required Values

An integer. See below

0	No limit to the number of skipped messages
1	Shutdown the e*Way on the first failed message.
>1	The actual number of skipped messages that will be tolerated.

5.1.4 Axion4 Retrieve

The parameters in this section specify the information required for the connection to access the external systems to retrieve messages to the axion4 gateway:

- Timeout Period

- Retrieve Channel
- Retrieve Mode
- Message Status
- Forward Retrieve API Acknowledgment
- Publish Single Topic Only

TimeoutPeriod

Description

Sets the timeout period in milliseconds the Retrieve API will block while waiting for a message.

Required Values

An integer between 0 and 864000. The default is 0. A value of 0 will result in using the default timeout specified in the Axion4 configuration file.

Retrieve Channel

Description

Specifies the channel from which the messages will be retrieved.

Required Values

TRADE, **QUERY**, or **REPORT**. The default is **TRADE**.

Retrieve Mode

Description

Specifies whether the status of the message should be changed after it has been retrieved. The message status is unchanged for BROWSE mode and is changed to PROCESSED For READ mode.

Required Values

BROWSE or **READ.**

Message Status

Description

Specifies the status of the messages that are to be retrieved.

Required Values

UNPROCESSED or **PROCESSED**.

Forward Retrieve API Acknowledgement

Description

Specifies whether to publish the result of the retrieve API call.

Required Values

Yes or **No**.

Publish Single Topic Only

Description

Specifies whether to publish the retrieve message to a single topic only (**axion4RetrieveOut**) or according to GSTPA message type.

Required Values

Yes or **No**. No causes results to be published according to GSTPA message type.

5.1.5 Axion4 Terminate

The parameters in this section specify the information required for the connection to access the terminate the connection to the axion4 gateway.

Terminate Mode

Description

Specifies whether to terminate the connection immediately or to send all waiting outgoing messages first.

Required Values

IMMEDIATE or **QUIESCE**. A 'TERMINATE IMMEDIATE' value causes the connection between the e*Way and axion4 to terminate immediately. A 'TERMINATE WITH QUIESCE' value causes all outbound messages in the axion4 database, that are awaiting delivery to be sent before the connection to the Gateway is closed.

Chapter 6

System Implementation

In this chapter we summarize the procedures required for implementing a working system incorporating the axion4 e*Way. Please refer to the *e*Gate Integrator User's Guide* for more information.

Note: *The e*Way must be run in the environment configured with Swiftnet profile, (filename/password) otherwise the API calls will fail.*

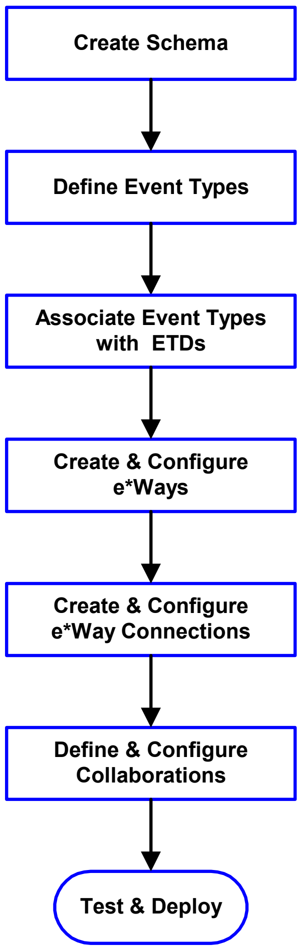
Important: *The user must monitor the Axion4 Gateway status manually to ensure that the configure API call succeeds, and the Gateway remains UP.*

6.1 Overview

This e*Way provides a specialized transport component for incorporation into an operational Schema. The schema also will contain Collaborations, linking different data or Event types, and Intelligent Queues. Typically, other e*Way types also will be used as components of the Schema.

Topics included in this chapter include:

6.1.1 Implementation Sequence


Create Schema
Define Event Types
Associate Event Types with ETDs
Create & Configure e*Ways
Create & Configure e*Way Connections
Define & Configure Collaborations
Test & Deploy

1 The first step is to create a new Schema—the subsequent steps will apply to this Schema (see **Creating a Schema** on page 42).

2 The second step is to define the Event Types you will be transporting and processing within the Schema (see **Creating Event Types** on page 42).

3 Next you need to associate the Event Types created in the previous step with Event Type Definitions (ETDs) derived from the applicable Business Rules (see **Generating Event Type Definitions** on page 43).

4 The fourth step is to create and configure the required e*Ways.

5 The fifth step is to configure the e*Way Connections.

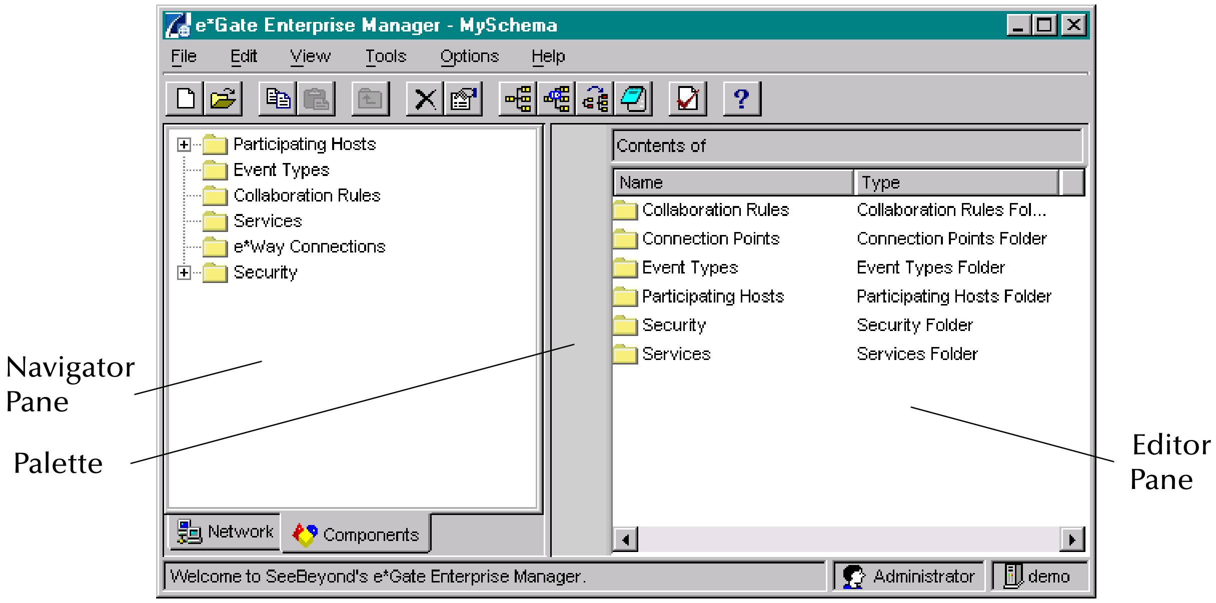
6 Next you need to define and configure the Collaborations between Event Types (see **Defining Collaborations and Collaboration Rules** on page 44).

7 Finally, you must test your Schema. Once you have verified that it is working correctly, you may deploy it to your production environment.

6.2 Using the e*Gate Enterprise Manager

First, here is a brief introduction to the e*Gate Enterprise Manager. The general features of the e*Gate Enterprise Manager window are shown in Figure 4.

Figure 4 e*Gate Enterprise Manager Window (Components View)



Name	Type
Collaboration Rules	Collaboration Rules Fol...
Connection Points	Connection Points Folder
Event Types	Event Types Folder
Participating Hosts	Participating Hosts Folder
Security	Security Folder
Services	Services Folder

Welcome to SeeBeyond's e*Gate Enterprise Manager.

Use the Navigator and Editor panes to view the e*Gate components. Note that you may only view components of a single schema at one time, and that all operations apply only to the current schema. Specialized command buttons (see Figure 5) appear in the Palette area of the window, depending on which levels of the Components Tree are open. For additional information, see the *e*Gate Integrator User's Guide*.

Figure 5 Setup Command Buttons

Button	Name	Location	Function
	Create e*Way	Palette	Creates a new e*Way
	Create e*Way Connection	Palette	Creates a new e*Way Connection I
	Create BOB	Palette	Creates a new Business Object Broker
	Create e*Insight Engine	Palette	Creates a new e*Insight Engine (only if e*Insight BPM is installed)

6.3 Creating a Schema

A schema is the structure that defines e*Gate system parameters and the relationships between components within the e*Gate system. Schemas can span multiple hosts.

Because all setup and configuration operations take place within an e*Gate schema, a new schema must be created, or an existing one must be started before using the system. Schemas store all their configuration parameters in the e*Gate Registry.

To create a new schema

1. Launch the e*Gate Enterprise Manager and log in as **Administrator** (or other user with equivalent privilege) on the appropriate Registry Host.
2. When the **Open Schema on Registry Host** dialog box appears, click **New**.
3. In the **Enter New Schema Name** text box, enter a name for the new schema (e.g., **TestSchema**), then click **Open**.

The Enterprise Manager will open the new schema.

4. From the **Options** menu, click on **Default Editor** and select **Java**.
5. Select the **Components** tab, found at the bottom of the Navigator pane of the e*Gate Enterprise Manager window (see **Figure 4 on page 41**).

The e*Gate Enterprise Manager then opens under your new schema name. You are now ready to begin creating the necessary components for this new schema.

Note: *From this point forward, all procedures should be performed while displaying the Components Navigator pane.*

6.4 Creating Event Types

Within e*Gate, messages and/or packages of data are defined as Events. Each Event must be categorized into a specific Event Type within the schema.

To define the Event Types

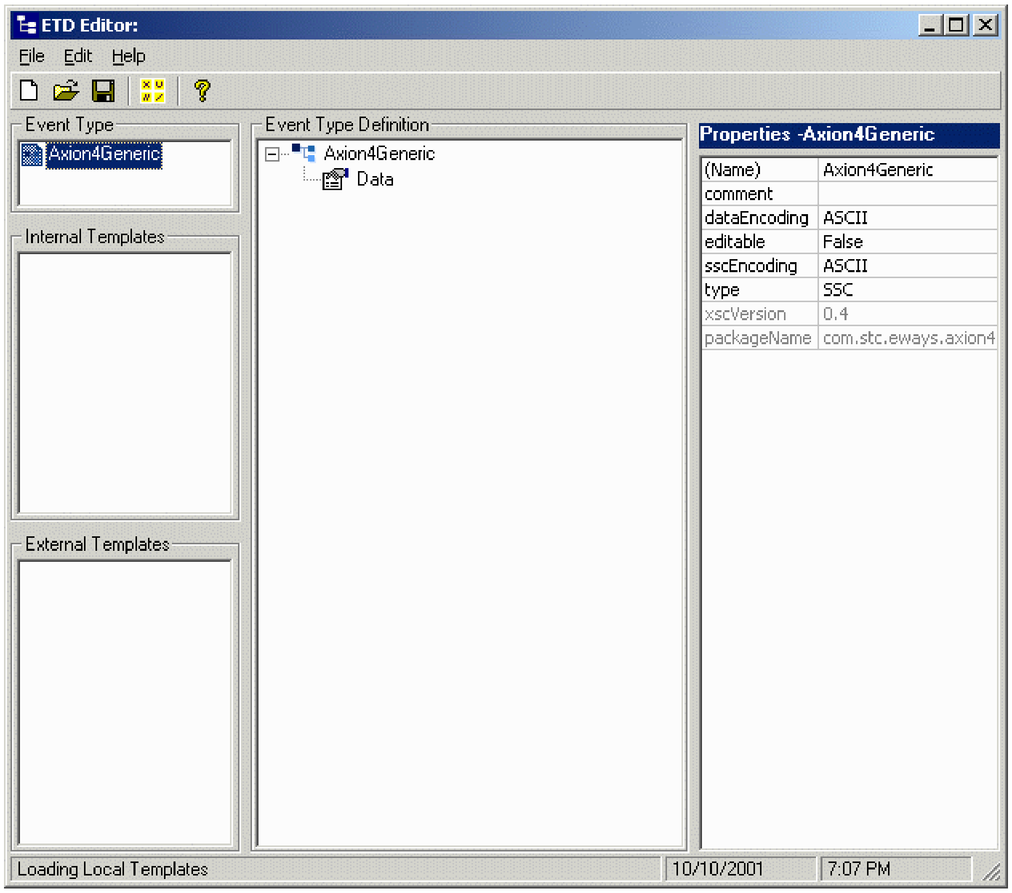
1. In the e*Gate Enterprise Manager's Navigator pane, select the **Event Types** folder.
2. On the Palette, click the **New Event Type** button.
3. In the **New Event Type Component** box, enter the name for the input Event Type and click **Apply**. Use this method to create all required Event Types, for example:
   - **blob (Generic Event)**
   - **GatewayAck**
4. After you have created the final Event Type, click **OK**.

6.5 Generating Event Type Definitions

As the name implies, an Event Type Definition (ETD) defines the structure of the Event Types employed in your Schema. Any one ETD can be associated with more than one Event Type within the schema. In the axion4 e*Way, ETDs are provided with the installation.

To use the pre-defined collaboration class files, the Axion4Generic.xsc event type is used for most instances. A graphical representation of the .xsc file appears below:

Figure 6 Axion4Generic.xsc



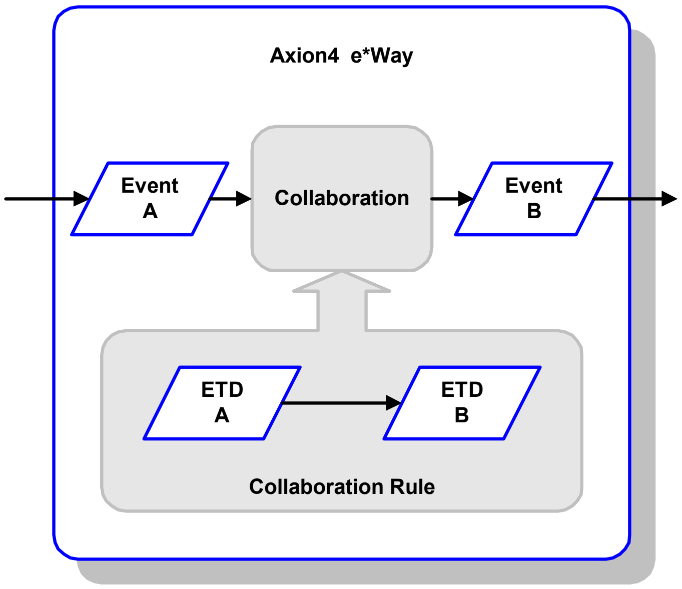
(Name)	Axion4Generic
comment	
dataEncoding	ASCII
editable	False
sscEncoding	ASCII
type	SSC
xscVersion	0.4
packageName	com.stc.eways.axion4

Loading Local Templates | 10/10/2001 | 7:07 PM

6.6 Defining Collaborations and Collaboration Rules

After you have located the required Event Type Definitions, you must define a Collaboration to transform the incoming Event into the desired outgoing Event. The Collaboration is driven by a Collaboration Rules script, which defines the relationship between the incoming and outgoing ETDs.

Figure 7 Collaborations



6.6.1 Using the Pre-defined Collaboration Rules

The current version of the axion4 e*Way provides .class files to be used as pre-defined Collaboration Rules. While is possible to create your own Collaboration Rules, in most cases the pre-defined Collaboration Rules should suffice.

In order to incorporate the .class files, the user must still create and associate the Collaboration Rule with the Collaborations. Do not re-compile the supplied Collaboration Class files.

Outbound Collaboration

During the Initialization state, for the first attempt to run the collaboration, or if the Gateway has gone down, the Collaboration tries to configure the Axion4 Gateway. If the attempt to configure returns successfully, or returns an indication that the Gateway is already configured, the Collaboration is ready to send messages. If the configure attempt returns other errors, the Collaboration will attempt to configure the Gateway up to three times, at which point, it publishes a “gatewayErr”, with the contents outputXML from the configure API call.

During messaging state, if the send API call returns an error code, the Collaboration checks the error code and performs the following:

1. If the error code indicates that the Gateway is not configured, then the Collaboration returns false, which rolls back the message, and the Collaboration returns to the initialization state.
2. If the error code indicates the input message has a duplicate message reference number, the Collaboration queries the message status, and concatenates the outputXML containing the error code, the original input message with the duplicate message status outputXML for publishing as a “gatewayErr”.
3. If the an error code other than the above mentioned is returned, the Collaboration publishes the concatenated data as “gatewayErr” with the contents of the outputXML containing the error along with the original message.

If the error is a result of the second or third possibility above, the failed message count is increased. If the configured failed message count is reached, an alert is sent to the control broker, requiring the user to manually shut down the e*Way.

If the send is successful, and no errors are returned, the commit API is called. If forward send API parameter is set to YES, then the Collaboration publishes a “gatewayAck” with the contents of the send outputXML.

The Instance Names that are implemented in the Outbound Collaboration are mandatory. See **“Outbound Collaboration Mapping” on page 49** for more information.

The following Instance Names that are implemented in the Outbound Collaboration are mandatory.

- axion4: Sends the data to the Gateway via the e*Way Connection
- axion4SendIn: Populated by the original message/data to be sent.
- gatewayAck: Receives the Send API outputXML from Gateway, if the Forward Send API is set to Yes.
- gatewayErr: Receives either the configure API outputXML with error code or send API concatenated with origianal message, or duplicate message status API concatenated with the send outputXML containing error code and the original message.

Inbound Collaboration

During the Initialization state, for the first attempt to run the collaboration, or if the Gateway has gone down, the Collaboration tries to configure the Axion4 Gateway. If the attempt to configure returns successfully, or returns an indication that the Gateway is already configured, the Collaboration is ready to receive messages. If the configure attempt returns other errors, the Collaboration will attempt to configure the Gateway up to three times, at which point, it publishes a “gatewayErr”, with the contents outputXML from the configure API call.

During messaging state, if no messages are received, the Collaboration waits for the specified sleep time-out and retries again. (Specified in the Event “get” interval, see the **“e*Way Connection Configuration” on page 33** for more information)

If the retrieve API call returns an error code, the Collaboration checks the error code and performs the following:

1. If the error code indicates that the Gateway is not configured, then the Collaboration returns false, which rolls back the message, and the Collaboration returns to the initialization state.

If the Collaboration successfully retrieves a message, the message is published as "axion4ReceiveOut" if the Publish Single Topic is set to Yes. If the Publish Single Topic is not set to Yes, the message is published as required by specified instance names that match the message type.

Any error or exception during this publication, is published as a "gatewayErr", and the Collaboration throws an exception.

Once the message is succelly retrieved and published, if the Forward Retrieve API parameter is set to Yes, the Collaboration publishes a "gatewayAck" with the contents of the retrieve API outputXML. After successful publication, the retrieve is committed to the database.

The following Instance Names that are implemented in the Inbound Collaboration are mandatory.

- axion4ReceiveIn: Retrieves the data from the Gateway via the e*Way Connection
- axion4ReceiveOut: Populated with retrieved data, if Publish Single Topic is set to Yes.
- gatewayAck: Receives the retrieve API outputXML from Gateway, if the Forward Retrieve API is set to Yes.
- gatewayErr: Receives either the configure API outputXML with error code or retrieve API outputXML with error code

Any additional Instance Names are optional, provided the Instance Name implemented corresponds with the settings in the Configuration and the GSTP Message Type. See **"Inbound Collaboration Mapping" on page 47** for more information.

If any exception occurs after the initial stage, but before the commit of an Inbound or Outbound Collaboration process, the rollback API is called.

6.6.2 Collaborations Rules

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

From the **Enterprise Manager Task Bar,** select **Options** and click **Default Editor**. For the beta Release, the default should be set to **Java**.

To create an Inbound to e*Gate Collaboration Rule Using the Pre-defined .class files

1. Select the Navigator's **Components** tab in the e*Gate Enterprise Manager.
2. In the **Navigator**, select the **Collaboration Rules** folder.
3. On the palette, click the icon.

4 Enter the name of the new Collaboration Rule, then click **OK.**

5 Select the new **Collaboration Rule**, then right-click to edit its properties.

6 On the **General** tab, in the **Service** box, select the **Java Collaboration Service**. The Collaboration Rules will use the Java Collaboration Service to manipulate Events or Event data.

7 In the **Initialization string** box, enter any required initialization string that the Collaboration Service may require. This field can be left blank.

8 Under the **Collaboration Rules** box, click **Find**.

9 Navigate to the following directory and select the file:

```
\\collaboration_rules\axion4\Axion4InboundCollaboration.class
```

10 Under the **Initialization File** box, click **Find**.

11 Navigate to the following directory and select the file:

```
\\collaboration_rules\axion4\Axion4InboundCollaboration.ctl
```

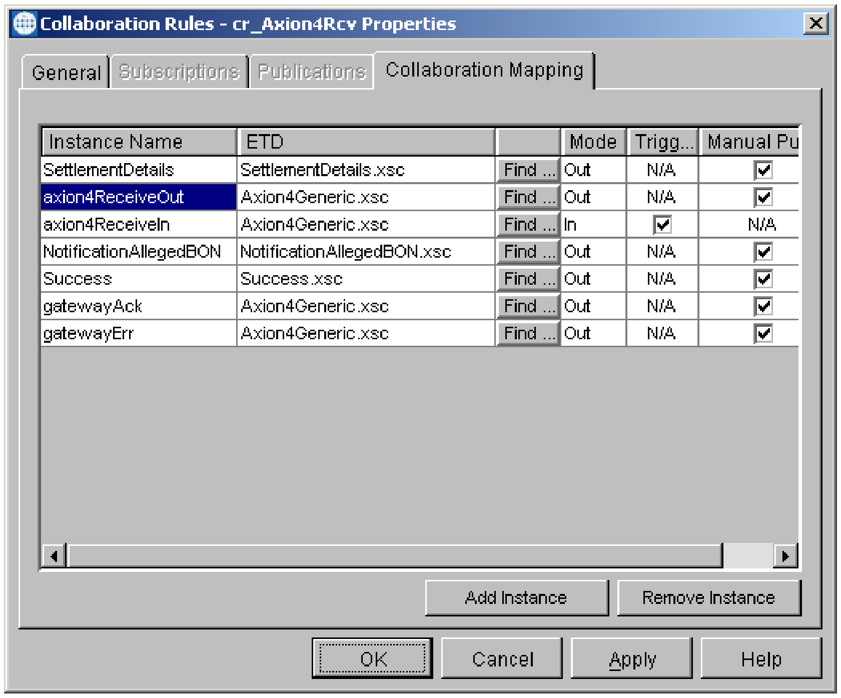
12 Select the **Collaboration Mapping** tab.

The Collaboration Rules - Collaboration Mapping Properties box opens:

13 Using the **Add Instance** button, create instances to coincide with the Event Types.

14 In the Instance Name column, create the following:

Figure 8 Inbound Collaboration Mapping



Instance Name	ETD		Mode	Trigg...	Manual Pu
SettlementDetails	SettlementDetails.xsc	Find ...	Out	N/A	✓
axion4ReceiveOut	Axion4Generic.xsc	Find ...	Out	N/A	✓
axion4ReceiveIn	Axion4Generic.xsc	Find ...	In	✓	N/A
NotificationAllegedBON	NotificationAllegedBON.xsc	Find ...	Out	N/A	✓
Success	Success.xsc	Find ...	Out	N/A	✓
gatewayAck	Axion4Generic.xsc	Find ...	Out	N/A	✓
gatewayErr	Axion4Generic.xsc	Find ...	Out	N/A	✓

In order to utilize the **Axion4Inbound.class** file, the conventions used above are mandatory. Both “NotificationAllegedBON” and “SettlementDetails” are optional. They are used if Publish Single Topic is set to Yes, and this is one of the expected message types. If an additional instance is desired for the Out Mode, the name of the instance must correspond to the name of the selected **.xsc** file without the extension.

Note: *If using multiple instances to receive data, the* ***Publish Single Topic Only*** *parameter must be set to* ***NO***. *If Publish Single Topic Only is set to YES, "axion4ReceiveOut" is required to return data.*

15 To locate the .xsc files, navigate to the following directory:

```
\\etd\Axion4\
```

In most cases **Axion4Generic.xsc** will suffice.

To create an Outbound from e*Gate Collaboration Rule Using the Pre-defined .class files

1 Select the Navigator's **Components** tab in the e*Gate Enterprise Manager.

2 In the **Navigator**, select the **Collaboration Rules** folder.

3 On the palette, click the icon.

4 Enter the name of the new Collaboration Rule, then click **OK.**

5 Select the new **Collaboration Rule**, then right-click to edit its properties.

6 On the **General** tab, in the **Service** box, select the **Java Collaboration Service**. The Collaboration Rules will use the Java Collaboration Service to manipulate Events or Event data.

7 In the **Initialization string** box, enter any required initialization string that the Collaboration Service may require. This field can be left blank.

8 Under the **Collaboration Rules** box, click **Find**.

9 Navigate to the following directory and select the file:

```
\\collaboration_rules\axion4\Axion4OutboundCollaboration.class
```

10 Under the **Initialization File** box, click **Find**.

11 Navigate to the following directory and select the file:

```
\\collaboration_rules\axion4\Axion4OutboundCollaboration.ctl
```

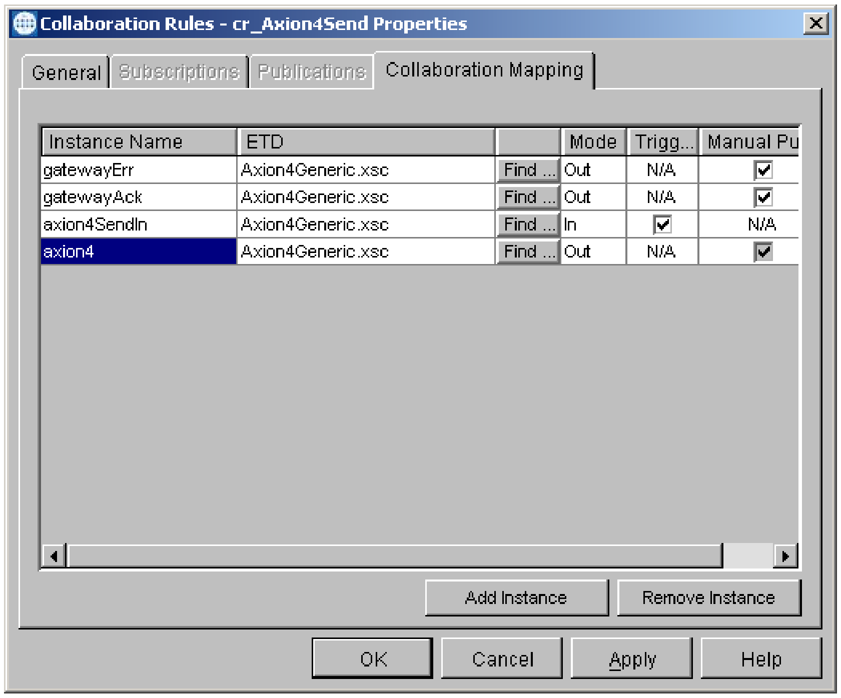
12 Select the **Collaboration Mapping** tab.

The Collaboration Rules - Collaboration Mapping Properties box opens:

13 Using the **Add Instance** button, create instances to coincide with the Event Types.

14 In the Instance Name column, create the following:

Figure 9 Outbound Collaboration Mapping



Instance Name	ETD		Mode	Trigg...	Manual Pu
gatewayErr	Axion4Generic.xsc	Find ...	Out	N/A	✓
gatewayAck	Axion4Generic.xsc	Find ...	Out	N/A	✓
axion4SendIn	Axion4Generic.xsc	Find ...	In	✓	N/A
axion4	Axion4Generic.xsc	Find ...	Out	N/A	✓

In order to utilize the Axion4Inbound.class file, the conventions used above are mandatory. In the case of Outbound Collaboration Rules, the In Mode instance must utilize **Axion4Generic.xsc**.

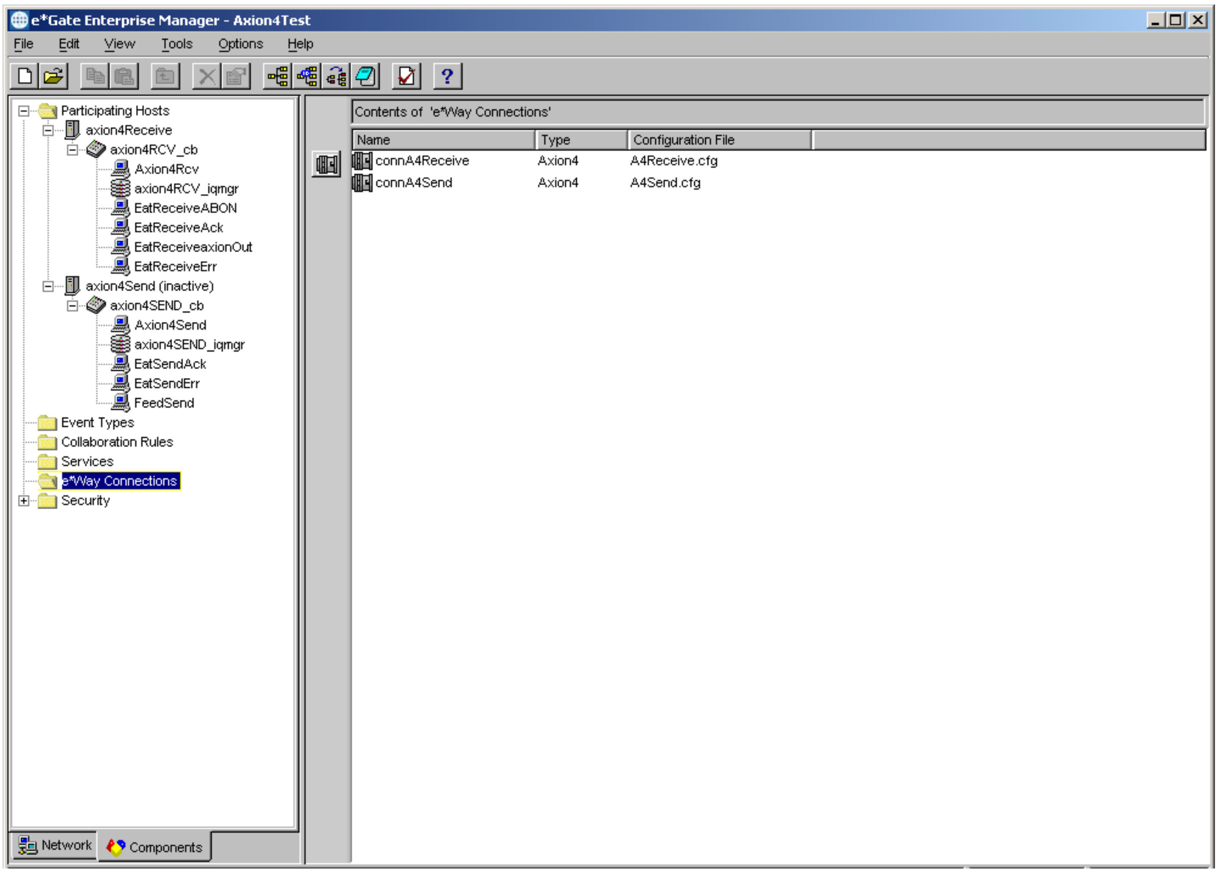
15 To locate the **.xsc** files, navigate to the following directory:

```
\\etd\Axion4\
```

In most cases **Axion4Generic.xsc** will suffice.

6.7 Sample Scenario

The sample schema can be found in the root directory of the Installation CD under \samples\ewaxion4. The schema depicted below demonstrates the use of the pre-defined .class files.



Contents of 'e*Way Connections'

Name	Type	Configuration File
connA4Receive	Axion4	A4Receive.cfg
connA4Send	Axion4	A4Send.cfg

The schema above utilizes two Participating Hosts, but this is not mandatory. The e*Ways have been separated to clearly distinguish between those that send data out to the Gateway and those that receive data from the Gateway. The schema consists of the following:

- Two Participating Hosts:
  - Axion4Rcv : Receives data from the Gateway
  - Axion4Send: Sends data to the Gateway
- Two Control Brokers:
  - Axion4Rcv_cb
  - Axion4Send_cb
- Two IQ Managers
  - rcv_iqmgr
  - send_iqmgr
- Two e*Way Connections (although one could be used)
  - cp_Axion4Rcv
  - cp_axion4Send

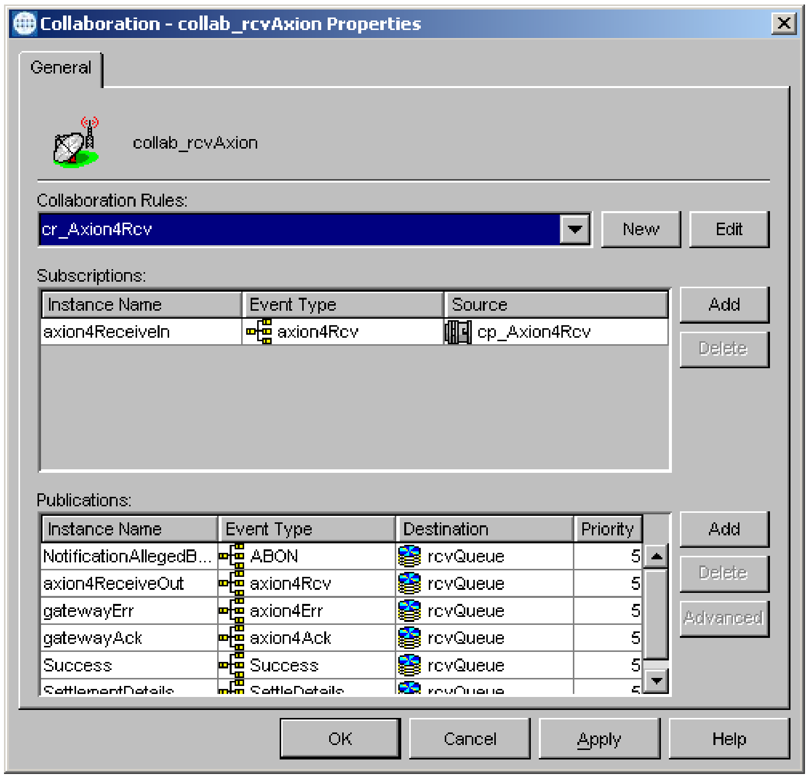
The Axion4Rcv_cb contains the following e*Ways:

- Axion4Rcv: Receives the actual data from the Gateway.

Executable File	Collaboration	Collaboration Rule	Collaboration Service
stceway.exe	collab_rcvAxion	Axion4InboundCollaboration.class	Java

The Collaboration properties are then created as follows:

Figure 10 collab_rcvAxion Collaboration



Collaboration - collab_rcvAxion Properties

General

collab_rcvAxion

Collaboration Rules: cr_Axion4Rcv

Subscriptions:

Instance Name	Event Type	Source
axion4ReceiveIn	axion4Rcv	cp_Axion4Rcv

Publications:

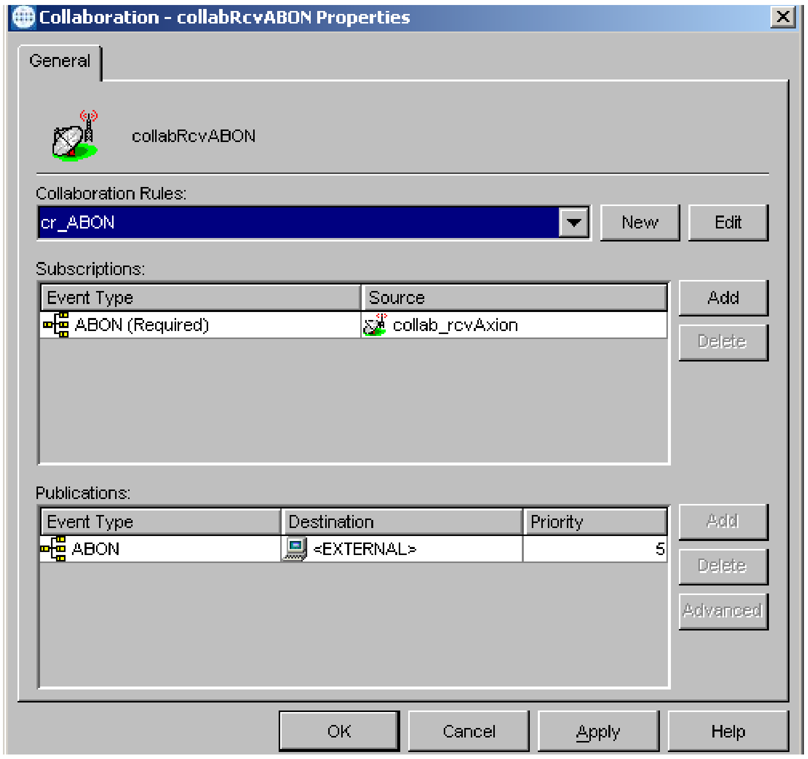
Instance Name	Event Type	Destination	Priority
NotificationAllegedB...	ABON	rcvQueue	5
axion4ReceiveOut	axion4Rcv	rcvQueue	5
gatewayErr	axion4Err	rcvQueue	5
gatewayAck	axion4Ack	rcvQueue	5
Success	Success	rcvQueue	5
SettlementDetails	SettleDetails	rcvQueue	5

- RcvABON: An optional File e*Way that receives specified notification from the Gateway.

Executable File	Collaboration	Collaboration Rule	Collaboration Service
stcewfile.exe	collabRcvABON	cr_ABON	Pass Through

The Collaboration properties are then created as follows:

Figure 11 collabRcvABON Collaboration


Collaboration - collabRcvABON Properties

General

collabRcvABON

Collaboration Rules: cr_ABON

Subscriptions:

Event Type	Source
ABON (Required)	collab_rcvAxion

Publications:

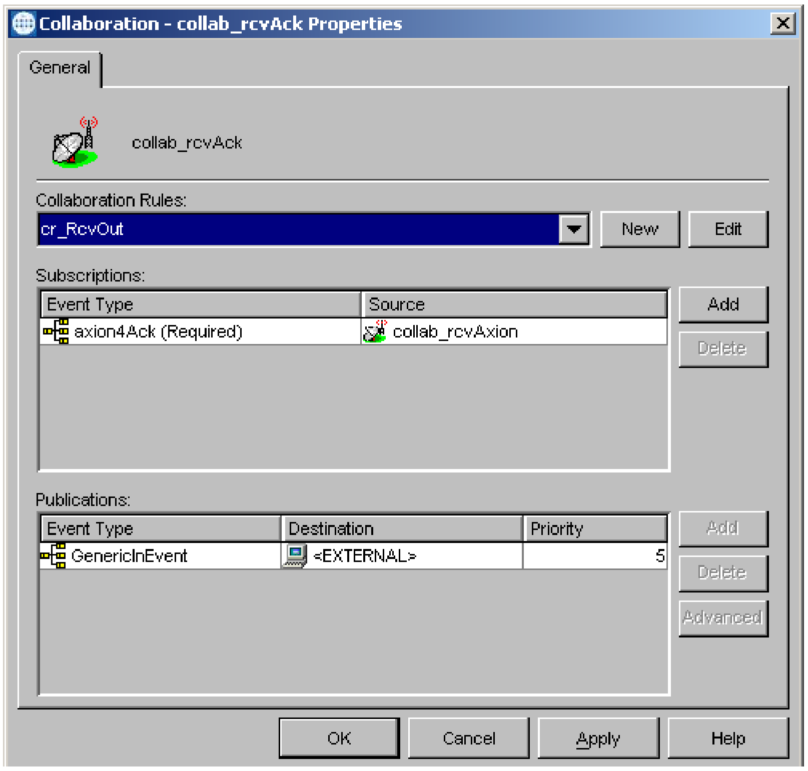
Event Type	Destination	Priority
ABON	<EXTERNAL>	5

- rcvAck: A File e*Way that receives acknowledgment information if the call is successful.

Executable File	Collaboration	Collaboration Rule	Collaboration Service
stcewfile.exe	collab_rcvAck	cr_RcvOut	Pass Through

The Collaboration properties are then created as follows:

Figure 12 collab_rcvAck Collaboration



Collaboration - collab_rcvAck Properties

General

collab_rcvAck

Collaboration Rules: cr_RcvOut

Subscriptions:

Event Type	Source
axion4Ack (Required)	collab_rcvAxion

Publications:

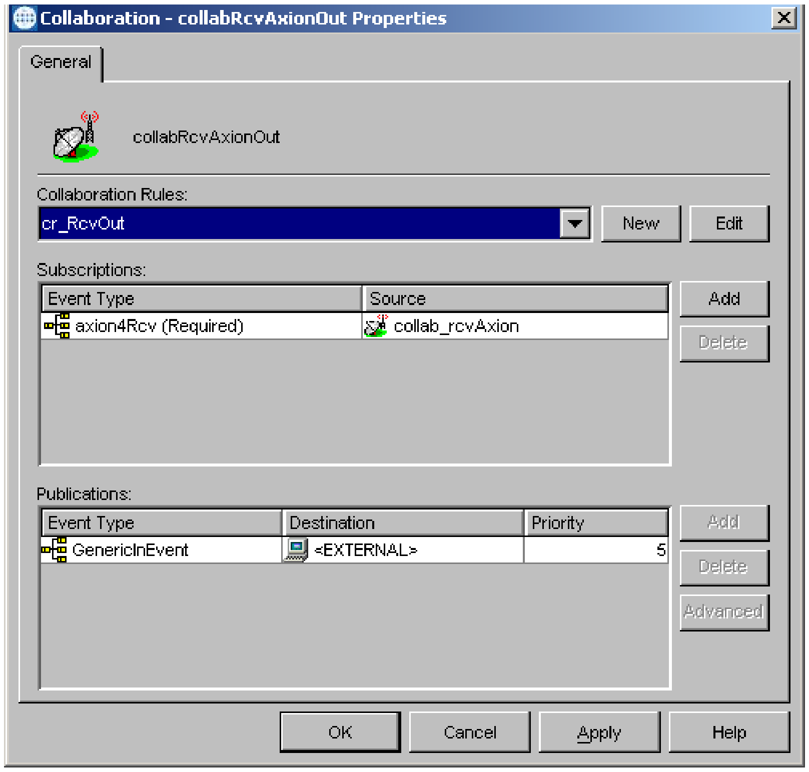
Event Type	Destination	Priority
GenericInEvent	<EXTERNAL>	5

- rcvAxionOut: A File e*Way

Executable File	Collaboration	Collaboration Rule	Collaboration Service
stcewfile.exe	collabRcvAxionOut	cr_RcvOut	Pass Through

The Collaboration properties are then created as follows:

Figure 13 collabRcvAxionOut Collaboration


Collaboration - collabRcvAxionOut Properties

General

collabRcvAxionOut

Collaboration Rules: cr_RcvOut

Subscriptions:

Event Type	Source
axion4Rcv (Required)	collab_rcvAxion

Publications:

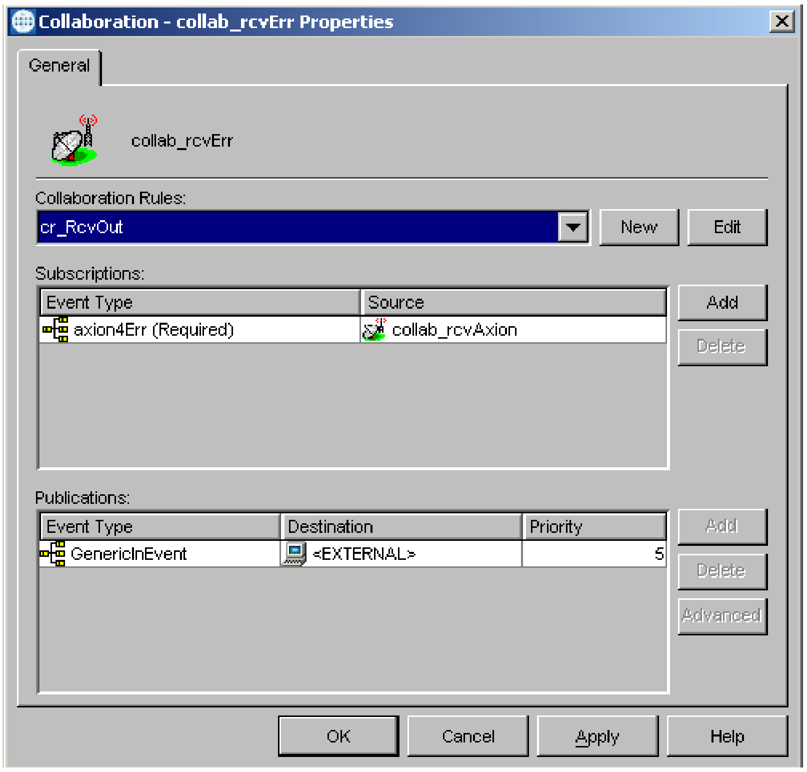
Event Type	Destination	Priority
GenericInEvent	<EXTERNAL>	5

- rcvErr: A File e*Way that receives error information if the call is unsuccessful.

Executable File	Collaboration	Collaboration Rule	Collaboration Service
stcewfile.exe	collab_rcvErr	cr_RcvOut	Pass Through

The Collaboration properties are then created as follows:

Figure 14 collab_rcvErr Collaboration


Collaboration - collab_rcvErr Properties

General

collab_rcvErr

Collaboration Rules: cr_RcvOut

Subscriptions:

Event Type	Source
axion4Err (Required)	collab_rcvAxion

Publications:

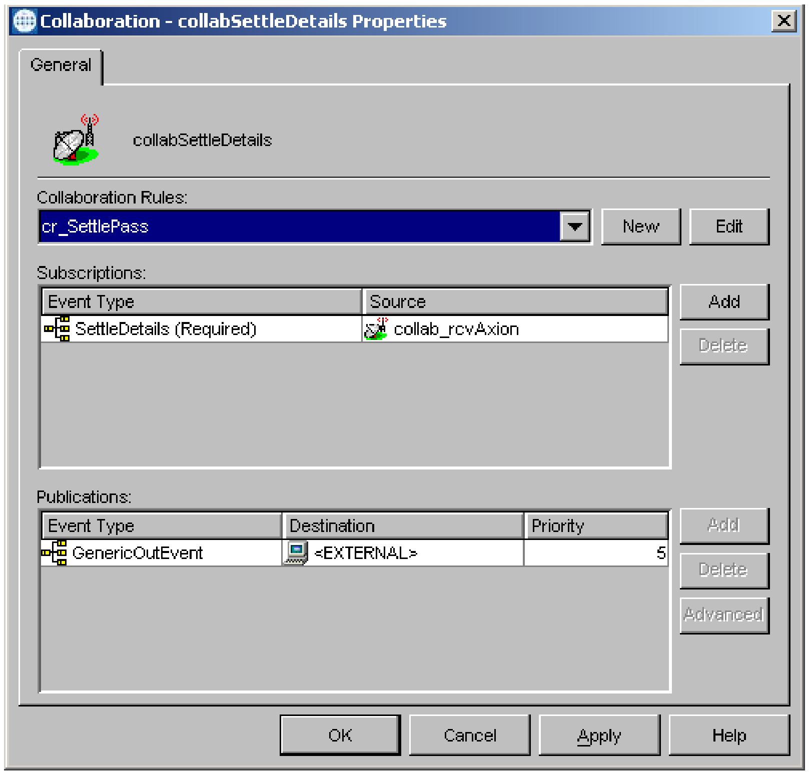
Event Type	Destination	Priority
GenericInEvent	<EXTERNAL>	5

- rcvSettleDetails: A File e*Way that receives error information if the call is unsuccessful.

Executable File	Collaboration	Collaboration Rule	Collaboration Service
stcewfile.exe	collabSettleDetails	cr_SettlePass	Pass Through

The Collaboration properties are then created as follows:

Figure 15 collabSettleDetails Collaboration



Collaboration - collabSettleDetails Properties

General

collabSettleDetails

Collaboration Rules: cr_SettlePass

Subscriptions:

Event Type	Source
SettleDetails (Required)	collab_rcvAxion

Publications:

Event Type	Destination	Priority
GenericOutEvent	<EXTERNAL>	5

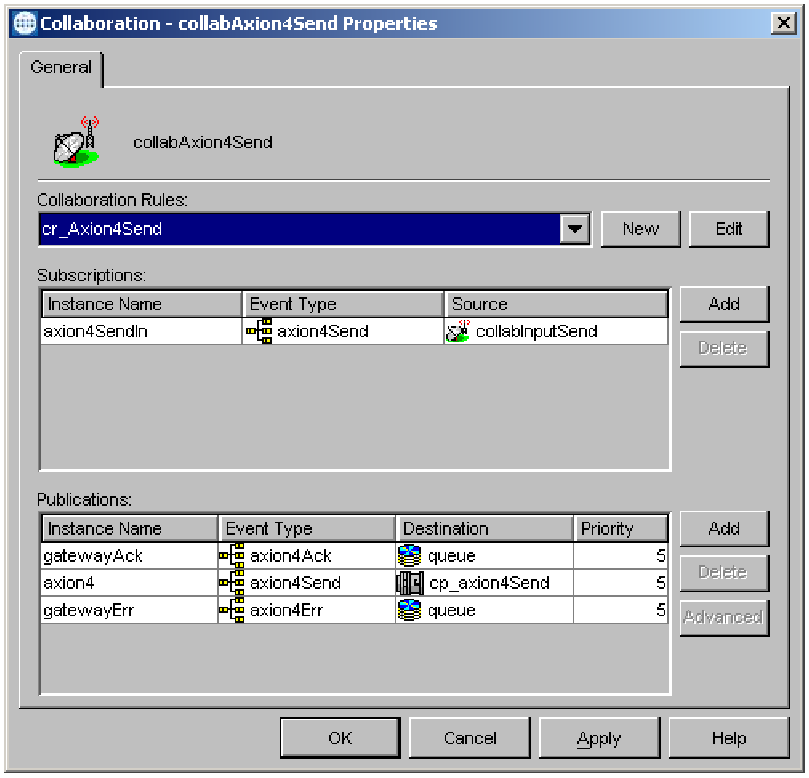
The axion4Send_cb contains the following e*Ways

- axion4Send: Sends the actual data to the Gateway.

Executable File	Collaboration	Collaboration Rule	Collaboration Service
stceway.exe	collabAxion4Send	Axion4OutboundCollaboration.class	Java

The Collaboration properties are then created as follows:

Figure 16 collabAxion4Send Collaboration



Collaboration - collabAxion4Send Properties

General

collabAxion4Send

Collaboration Rules: cr_Axion4Send

Subscriptions:

Instance Name	Event Type	Source
axion4SendIn	axion4Send	collabInputSend

Publications:

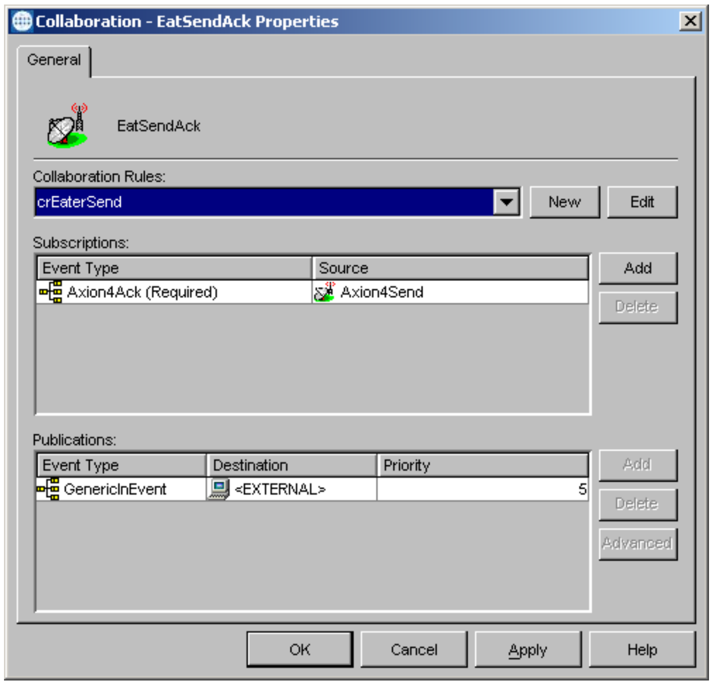
Instance Name	Event Type	Destination	Priority
gatewayAck	axion4Ack	queue	5
axion4	axion4Send	cp_axion4Send	5
gatewayErr	axion4Err	queue	5

- inputSend: A File e*Way that receives acknowledgment information if the call is successful.

Executable File	Collaboration	Collaboration Rule	Collaboration Service
stcewfile.exe	collabInputSend	cd_PassThruIn	Pass Through

The Collaboration properties are then created as follows:

Figure 17 collabInputSend Collaboration



Collaboration - EatSendAck Properties

General

EatSendAck

Collaboration Rules: crEaterSend

Subscriptions:

Event Type	Source
Axion4Ack (Required)	Axion4Send

Publications:

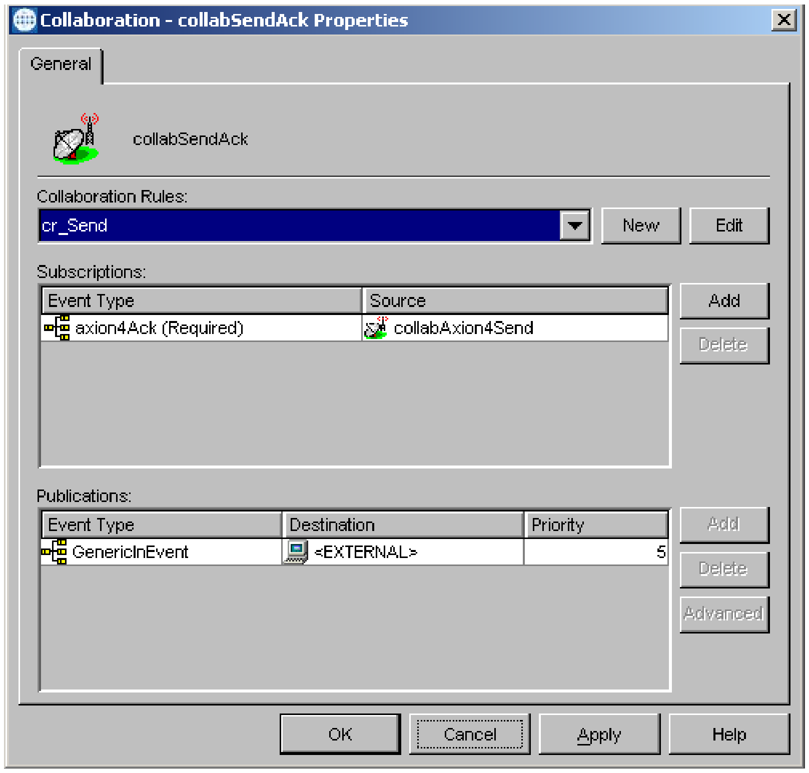
Event Type	Destination	Priority
GenericInEvent	<EXTERNAL>	5

- sendAck: A File e*Way that receives error information if the call is unsuccessful.

Executable File	Collaboration	Collaboration Rule	Collaboration Service
stcewfile.exe	collabSendAck	cr_Send	Pass Through

The Collaboration properties are then created as follows:

Figure 18 collabSendAck Collaboration



collabSendAck

Collaboration Rules: cr_Send

Subscriptions:

Event Type	Source
axion4Ack (Required)	collabAxion4Send

Publications:

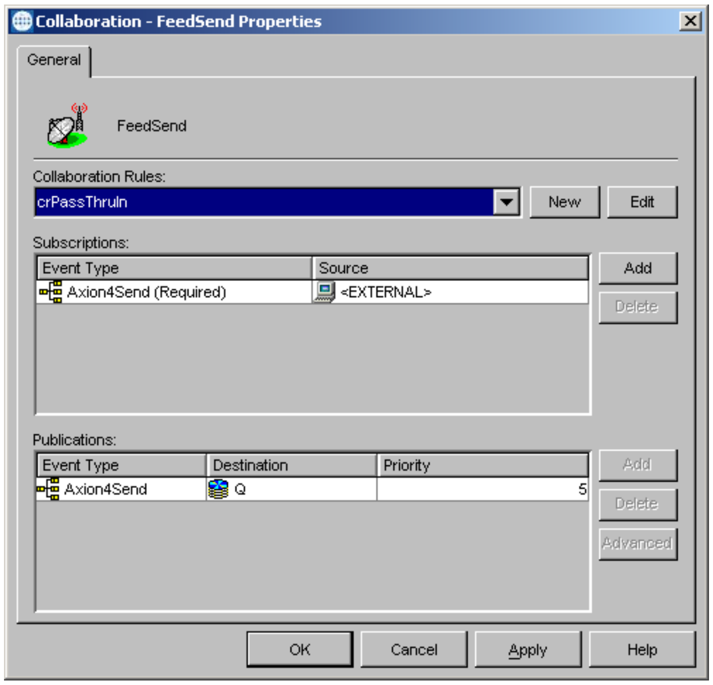
Event Type	Destination	Priority
GenericInEvent	<EXTERNAL>	5

- sendErr: A File e*Way grabs the file from some specified location and passes it into e*Gate to be sent to the Gateway.

Executable File	Collaboration	Collaboration Rule	Collaboration Service
stcewfile.exe	collabSendErr	cr_Send	Pass Through

The Collaboration properties are then created as follows:

Figure 19 collabSendErr Collaboration



Subscriptions:

Event Type	Source
Axion4Send (Required)	<EXTERNAL>

Publications:

Event Type	Destination	Priority
Axion4Send	Q	5

Unlike with the Collaboration Mapping required by the Collaboration Rules, the above naming conventions are not mandatory. They have been used for the sake of simplicity.

It must be remembered that unless the Subscription/Publication is associated correctly, the data passage will fail.

It is not relevant what naming conventions are used for the Event Types as long as the correct associations are made when creating the Instance Names and the ETD associations.

For more information, see the *e*Gate Integrator User's Guide*.

6.8 Importing a Sample Schema

The sample schema can be found on the Installation CD on the <root directory> under:

```
samples\ewaxion4
```

eGate Schema Set up

Axion4.zip

```
-------------------------------------------------------------------
```

1 Install the Axion4 e*Way add on.

2 Import the sample schema

eGate 4.5.1 and higher

1 In eGate Enterprise Manager, go to the File menu and select "Import Definitions from File", select Schema in the Import wizard, select Axion4.zip for Schema File Name. Axion4.zip is the file supplied in this sample directory.

pre-eGate 4.5.1

1 Extract the include Axion4.zip into a schemas repository on the registry server:

```
e.g.: egate\server\registry\repository\Axion4\.
```

2 Using stcregutil.exe, import the Axion4.exp into this new schema

```
stcregutil.exe -rh localhost -rs Axion4 -un Administrator -up STC -i
Axion4.exp
```

3 Using stcregutil.exe, import the Axion4.ctl into the new schema

```
stcregutil.exe -rh localhost -rs Axion4 -un Administrator -up STC -fc
. -ctl Xion4.ctl
```

4 Register control brokers for this new schema:

```
      stccb.exe -ln axion4Receive_cb -rh localhost -rs Axion4 -un
Administrator -up STC -sm

stccb.exe -ln axion4Send_cb -rh localhost -rs Axion4 -un
Administrator -up STC -sm
```

5 Start the newly registered cbs: from control panel, double-click services, find "eGate Control Broker (axion4Receive, axion4Send)", click on it and click on start.

Chapter 7

axion4 gateway e*Way Methods

The methods discussed in this chapter are contained within the java package **com.stc.eways.axion4**. This package is located in the **stcaxion4.jar file**.

The exception (STCxception) is located in the package **com.stc.eways.exception**. This package is currently located in the jar file **stcexception.jar**.

The axion4 gateway e*Way Java class contains the following functions:

7.1 axion4 Methods

axion4Configure

Syntax

```
public boolean axion4Configure(String filename, String password)
throws STCException
public bool axion4Configure()throws STCException
```

Description

Configures the Gateway with specified filename and password. The information for outbound processes for Trade, Query, and Report are provided by the configuration file.

Configures the Gateway and creates a connection to the TFM.

Parameters

Name	Type	Description
filename	String	The fully qualified path of the SNL security profile file.
password	String	The password required to access the SNL security profile file.

Return Values

Boolean

Returns **true** when successful; otherwise, returns **false** when an error occurs.

Throws

com.stc.eways.exception.STCException indicating that an exception was thrown and measures must be taken to catch the indicated exception.

axion4MsgStatus

Syntax

```
public boolean axion4MsgStatus(String messageReferenceNumber) throws
STCException
```

Description

Queries the status of a message with a specified reference number.

Parameters

Name	Type	Description
messageReferenceNumber	String	The Message Reference Number of a message.

Return Values

Boolean

Returns **true** when successful; otherwise, returns **false** when an error occurs.

Throws

com.stc.eways.exception.STCException indicating that an exception was thrown and measures must be taken to catch the indicated exception.

axion4Retrieve

Syntax

```
public boolean axion4Retrieve() throws STCException
public boolean axion4Retrieve(String messageReferenceNumber) throws
STCException
```

Description

Retrieves the next message from the Gateway. The retrieve channel, mode and TimeOutPeriod are provided within the configuration file. AutoCommit is set to NO. Use getOutputXML() to retrieve the output XML containing the next message following this method call. See getOutputXML()

Parameters

Name	Type	Description
messageReferenceNumber	String	The Message Reference Number for the message to be retrieved.

Return Values

boolean

Returns true if successful, otherwise, returns false.

Throws

com.stc.eways.exception.STCException indicating that an exception was thrown and measures must be taken to catch the indicated exception.

axion4Send

Syntax

```
public boolean axion4Send(String message) throws STCException
```

Description

Sends a message to the Gateway. The message priority is provided in the configuration file.

Parameters

Name	Type	Description
message	string	The outbound message string.

Return Values

Boolean

Returns **true** when successful; otherwise, returns **false** when an error occurs.

Throws

com.stc.eways.exception.STCException indicating that an exception was thrown and measures must be taken to catch the indicated exception.

axion4TerminateConnection

Syntax

```
public boolean axion4TerminateConnection() throws STCException
```

Description

Closes the connection between the Gateway and the TFM. The Terminate mode is provided in the configuration file.

Parameters

None.

Return Values

Boolean
Returns **true** when successful; otherwise, returns **false** when an error occurs.

Throws

com.stc.eways.exception.STCException indicates that an exception was thrown and measures must be taken to catch the indicated exception.

commitRetrieve

Syntax

```
public boolean commitRetrieve() throws STCException
```

Description

Commits previous retrieve API call(s).

Parameters

None.

Return Values

Boolean
Returns **true** when successful; otherwise, returns **false** when an error occurs.

Throws

com.stc.eways.exception.STCException indicating that an exception was thrown and measures must be taken to catch the indicated exception.

commitSend

Syntax

```
public boolean commitSend()throws STCException
```

Description

Commits the previous send API call(s).

Parameters

None.

Return Values

Boolean
Returns **true** when successful; otherwise, returns **false** when an error occurs.

Throws

com.stc.eways.exception.STCException indicating that an exception was thrown and measures must be taken to catch the indicated exception.

commitSendAndRetireve

Syntax

```
public boolean commitSendAndRetrieve()throws STCException
```

Description

Commits the previous send and retrieve API call(s).

Parameters

None.

Return Values

Boolean
Returns **true** when successful; otherwise, returns **false** when an error occurs.

Throws

com.stc.eways.exception.STCException indicating that an exception was thrown and measures must be taken to catch the indicated exception.

getOutputXML

Syntax

```
public boolean getOutputXML()
```

Description

Retrieves th eoutput XML string of an API call, such as retrieve and messageStatus.

Parameters

None.

Return Values

String
Returns the output XML string for the previous API call.

Throws

None.

rollbackRetrieve

Syntax

```
public boolean rollbackRetrieve()throws STCException
```

Description

Rolls back the the previous retrieve API call(s).

Parameters

None.

Return Values

Boolean
Returns **true** when successful; otherwise, returns **false** when an error occurs.

Throws

com.stc.eways.exception.STCException indicating that an exception was thrown and measures must be taken to catch the indicated exception.

rollbackSend

Syntax

```
public boolean rollbackSend()throws STCException
```

Description

Rolls back the the previous send API call(s).

Parameters

None.

Return Values

Boolean
Returns **true** when successful; otherwise, returns **false** when an error occurs.

Throws

com.stc.eways.exception.STCException indicating that an exception was thrown and measures must be taken to catch the indicated exception.

rollbackSendAndRetrieve

Syntax

```
public boolean rollbackSendAndRetrieve()throws STCException
```

Description

Rolls back the the previous send and retrieve API call(s).

Parameters

None.

Return Values

Boolean
Returns **true** when successful; otherwise, returns **false** when an error occurs.

Throws

com.stc.eways.exception.STCException indicating that an exception was thrown and measures must be taken to catch the indicated exception.

Index