Installation and Configuration

Cyclone Activator, BEA WebLogic Edition Series 5.4

12/14/2005

Third-party embedded products

Copyright credits and additional information

Depending on the specific Cyclone Commerce product configuration purchased, the product may contain some or all of the following third party files.

The vendors listed below in each case: (i) have provided such files "AS IS" and have affirmatively disclaimed any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose; and (ii) have not agreed to incur (and have affirmatively disclaimed) any liability (including but not limited to direct, indirect, incidental, special, exemplary or consequential damages of any kind) to any Cyclone Commerce customers or partners or other parties relating to the listed files, under any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of such files, even if the vendor had been advised of the possibility of such damage.

- This product includes software developed by the Apache Software Foundation (http://www.apache.org/), including the following files (see *ADDITIONAL NOTES RE APACHE FILES* at the end of the list of Apache files):
- *axis.jar*: this file is copyright 1999-2003 The Apache Software Foundation.
- *derby.jar*, *derbytools.jar*: these files are copyright 2004-2005 The Apache Software Foundation.
- *j2ssh.jar*: this file is copyright 2002-2003 Lee David Painter and Contributor, and distributable under the Apache Software Foundation license.
- *mx4j.jar*: this file is copyright 2001-2004 by MX4J contributors (http://mx4j.sourceforge.net/), and distributable under the Apache Software Foundation license.
- *ebxmlrr-bindings.jar; ebxmlrr-common.jar; jaxr-ebxml.jar*: these files are copyright 2000 The Apache Software Foundation. Portions of this Apache software may be based upon public domain software originally written at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign.
- *jasper-compiler.jar; jasper-runtime.jar; log4j.jar*: these files are copyright 2000-2004 The Apache Software Foundation. Portions of this Apache software may be

Trademarks

Third-party web sites and information

Contents

1 System requirements

The following topics provide the hardware and software requirements for running Cyclone Activator, BEA WebLogic Edition.

In addition, we strongly recommend that you read the release notes in the readme file on the installation CD for supplemental information about system requirements and installation.

Concepts

Hardware

The following topics outline the minimum hardware requirements.

Windows and UNIX computers

These are the hardware requirements for Windows and UNIX computers.

- 800 MHz or faster Pentium III-class processor (Windows only)
- Minimum 512 megabytes of Random Access Memory (RAM), but 1 gigabyte recommended
- 150 MB disk space for Cyclone Activator software
- 200-400 MB for data storage, but possibly more (see Temp directory requirement)
- SVGA monitor
- CD drive (for installation)
- TCP/IP network interface
- Local area network (LAN) card (Windows only)

Temp directory requirement

The temp directory of the computer running the trading engine server must have enough space available to handle the largest messages traded. As a rule of thumb, the temp directory space should be five times larger than the largest message times the number of messages processed concurrently. For example, if the largest message is 1 gigabyte and up to 5 messages of that size may be processed concurrently (inbound or outbound), the temp directory should have 25 gigabytes of available space. The following is the formula for this example:

1 GB largest message * 5 * 5 concurrent messages = 25 GB temp directory

When estimating the number of documents processed concurrently, we recommend estimating high rather than low. Your estimate depends on the number of integration and delivery exchanges, how the exchanges are configured, and the number of inbound and outbound documents going through the exchanges. Another way is to base the estimate on the expected throughput and how long it takes to process each document. For example, if the system is expected to process 100 documents a minute and it takes an average of 15 seconds to process one document, then 25 is the number processed concurrently.

E-mail server

The trading engine needs access to an external SMTP server if you plan on sending business messages via SMTP/POP or messages about system events via e-mail.

Software

The following topics outline the minimum software requirements.

Microsoft Windows

Cyclone Activator supports the following Windows operating systems.

- Windows XP with Service Pack 1 or later
- Windows Server 2003

You must have administrator rights to successfully install Cyclone Activator.

If you plan to run Cyclone Activator as a Windows service, we highly recommend creating a user account solely for starting the service upon log-on for Cyclone Activator and related applications (for example, AS2 Accelerator, aXML Agent). This user account should be granted all Windows administrator rights. The account should be fully dedicated to Cyclone Activator and not be used by any other user.

The dedicated user account is recommended because if a user logs on to a Windows computer, whatever applications were started are terminated by Windows when that user logs out.

After installing Cyclone Activator or related application as a Windows service, open the properties for the service and change the default setting of “Local System account” to “This account.” Then type the name and password of the dedicated user account. When the Windows computer starts up, the service will start as that special log-in account. So long as no one manually logs on to the Windows computer as that special log-in account name, the application will not be aborted when another user logs out of the computer.

The application installer provides the option to set up the Windows service under a local system account or a domain\user account. Choose a domain account if the service needs to access your network or other resources requiring user permissions beyond your local account. If unsure, ask your network administrator. Type a domain\user and password only for a domain account; you do not have to do so for a local account. If you decline to set up the Windows service at installation, you can set up the service later.

Even if you do not run Cyclone Activator or related applications as a Windows service, a dedicated user account for the Windows computer is still recommended. When the dedicated account is used to start the Windows computer and then Cyclone Activator is started manually, make sure the dedicated user never logs out to ensure the application will run continuously.

UNIX

Cyclone Activator supports the following UNIX operating systems.

- AIX 5.3 (see note)
- HP-UX 11i
- Red Hat Enterprise Linux 3.1 and 4
- Solaris 9 and 10

Note: When version 5.4 of Cyclone Activator was released, IBM’s JRE 1.5 for AIX was in beta. So support of AIX is pending. A version for AIX is planned once IBM releases the JRE. Meantime, AIX users may continue to use 5.3.3.

OS patches

Patches for some operating systems are required to support Java technology. We recommend checking the support web sites of OS vendors to download up-to-date patches for UNIX operating systems.

Requirement for users of HP-UX

Default values in HP-UX 11i for threads and other processes are too low for a Java application like Cyclone Activator. We recommend using the SAM tool to adjust kernel parameters to the values in the following table. We recommend that you involve your organization's UNIX administrator in making these changes.

Parameter	New value	Description
max_thread_proc	1024	Maximum number of threads allowed in each process.
maxfiles	4096	Soft file limit per process.
maxfiles_lim	6144	Hard file limit per process.
nfile	9968	Maximum number of open files.
nflocks	8192	Total number of file locks for open files system-wide.
ninode	2492	Maximum number of open inodes.
nkthread	3635	Total number of kernel threads available.
nproc	2068	Maximum number of proc table entries.

Database

Cyclone Activator comes with the Apache Derby open source database. The database is part of the installation and runs without configuration when the server is started.

For information about Derby go to http://db.apache.org/derby/index.html.

When using Derby, there is a chance the following error message may display, sometimes many times in a short period: "Share violation: another process may be using the file." This error occurs when a process other than the trading engine (most likely virus checking software) tries to access the

Derby database files. The error is harmless and does not affect operation of the trading engine. To prevent the error from displaying, set virus software to omit checking the database directory and files. The directory is at [install directory]\[build number]\corelib\db\derby. Do not move or try to change any of the files in this directory.

Internet connection

A persistent Internet connection is required for Cyclone Activator to exchange messages with partners. This means the connection should always be “on” and not a dial-up connection. The Internet connection also needs a static IP address, never a dynamic IP address common to dial-up connections.

For HTTP connections, it is likely that your partners need to go through a proxy server to connect to you. Ask your Internet service provider for the IP address or fully qualified domain name (FQDN) of the proxy server that a partner needs in order to connect to you over the Internet. Does the proxy server always present the same static IP address to the partner who needs to connect to you? If using an FQDN on the proxy server, does it always resolve to a static IP address? A dynamic IP address is unacceptable since partners who must connect to you would never know what IP address to use from one connection to the next.

Internet browser

The browser-based user interface and online help support Microsoft Internet Explorer 6 or later and Mozilla Firefox 1.0 or later.

If you experience display problems while using the online help (for example, table of contents links do not match pages), do the following.

- For IE, delete the browser’s temporary Internet files (select **Tools > Internet Options,** General tab). Restart the browser and try again.
- For Firefox, clear the cache (select **Tools** > **Options** and then select **Privacy** and click **Clear** next to **Cache**).

Pop-up blocking software warning

Pop-up blocking software for your browser may interfere with this product. You may want to disable or uninstall such software.

Browser requires JRE plug-in

The browser must have J2SE Java Runtime Environment (JRE) 1.5. This is required to run various applets within the user interface (for example, the server monitor and statistics monitor).

Do the following to check the browser for the plug-in:

Internet Explorer

Select **Tools** > **Internet Options** to open the Internet Options window. Select the **Advanced** tab. Scroll down the list of settings and look for a category named Java (Sun). Under this might be a setting that is the same or similar to the following line:

Use JRE 1.5.0_05 for <applet> (requires restart)

If present, select the check box for this entry if not already selected. Click **OK** to close the window and save your change and restart Internet Explorer.

Mozilla Firefox

Type **about:plugins** in the address field and press **Enter**. Scroll down the installed plug-ins page and check the version of the latest Java plug-in.

If you need JRE 1.5.0_05 or later, go to the following URL and download and install the JRE: http://java.sun.com/j2se/1.5.0/download.jsp

E-mail clients

If you plan to exchange messages with partners who use an e-mail client application rather than a trading engine application such as Cyclone Activator, partners are advised to use Microsoft Outlook 2002 or later.

E-mail service provider

If you plan to use e-mail as a transport for exchanging documents with partners, we recommend subscribing to a reputable Internet service provider with high-quality e-mail service. Do not use any of the free e-mail services available on the Internet. Although these services are suitable for personal messages, they are not acceptable for business-to-business e-commerce transactions. Such services have shortcomings related to file sizes and non-standard MIME headers that are fatal to document trading.

Remote access

Clients operating remotely require access to Cyclone Activator server. For details see UI usage with proxy servers on page 69.

Adobe Acrobat

Adobe Acrobat Reader 5 or later is required to view and print Cyclone Activator user documentation that is provided in portable document format (PDF) files. Acrobat Reader is available free from Adobe Systems Inc., www.adobe.com.

Port assignments and possible conflicts

Some transports and database types have default port assignments in Cyclone Activator. A default port assignment could conflict with a port already in use on your system. You can execute a command to check for possible conflicts. You can use the command before installing Cyclone Activator to eliminate possible port conflicts or after installing if you encounter a problem.

The following table lists the default port assignments in Cyclone Activator. The table says where you can change a port number.

Table 1 - Port assignments

Description	Default port	Where to change it
Derby JDBC driver	0	..\conf\datastoreconfig.xml
RMI	2010	..\bin\startServer.cmd
Control node listener for Server Monitor connections	3434	Not configurable.
Embedded SMTP	4025	This port, which the trading engine uses for an embedded server, can be changed by clicking **Configure the global embedded SMTP server** on the main trading configuration page in the user interface.

Table 1 - Port assignments

Description	Default port	Where to change it
Embedded HTTP or HTTPS	4080	This port, which the trading engine uses for an embedded server, can be changed by clicking **Configure the global embedded HTTP server** on the main trading configuration page in the user interface.
Embedded web services API server	5080	This port, which the trading engine uses for an embedded server, can be changed by clicking **Configure the global embedded web services API server** on the main trading configuration page in the user interface.
User interface HTTP	6080	This port is used to access the user interface in a browser. This can be changed in ..\conf\startup.xml.
Clusterbus	47001	Not configurable. If 47001 is in use, 47002 is tried and so on up to 47556.

If there is a port conflict, you can free the port for use by Cyclone Activator. Alternately, you can configure Cyclone Activator to use an inactive port above 1024 that is not listed in the output of the netstat command.

You can use the netstat command to generate a list of ports in use on your system. The command is executed in the following ways.

Windows

In a command prompt or DOS window, type **netstat -a -n** or **netstat - an** to display a list of ports in use. You can instead type **netstat -a -n | more** to page through the list.

UNIX

On a command line, type **netstat -a -n** or **netstat - an** to display a list of ports in use. Or, to find whether a specific port is in use, type **netstat -a | grep [port number]**.

Security considerations

To ensure the integrity of data processed by Cyclone Activator, we recommend that you adhere to the following security measures in addition to your company's own security policies. Although the risks are possibly remote, failure to institute minimum security measures may result in compromised data.

1. Install Cyclone Activator in the data layer behind a firewall and not in an area unprotected from exposure to the Internet.
2. Do not view a binary document that has been received by the trading engine without first scanning the document for viruses.
3. Institute a policy for periodically changing the password for accessing Cyclone Activator.
4. Control access to the computer running Cyclone Activator to authorized users.
5. If you use an external database on a different computer than the one running Cyclone Activator, control access to the database computer to authorized users.
6. If you manually distribute your certificate to partners, do so via a secure means. Encourage your partners to do likewise.

About the user license file

The functionality you are licensed to use is controlled by a file named license.xml. You must have a license file to install the application. After installing, the file is stored in [install directory]\[build number]\conf.

Do not move, rename, or delete the license.xml file. Any attempt to change the contents will make your system inoperable. The file is hashed and signed to protect it from tampering.

If you receive a new license.xml file, copy over the old file or replace it and restart the server. For example, you may receive a new license to replace one that has expired.

❖ ❖ ❖

2 Installation

You can install Cyclone Activator, BEA WebLogic Edition on computers that meet the hardware and software criteria described in System requirements on page 1.

The following topics describe how to install Cyclone Activator on all supported operating systems.

Concepts

- Installation outline
- Before installing on page 12

Procedure

- Installation procedure on page 13
- Start the server on Windows on page 19
- Start the server on UNIX on page 20
- Run as a Windows service on page 20
- Open the user interface on page 22
- Configuring external SMTP server on page 24
- Stop the server on page 24
- Uninstalling on page 24

Installation outline

The following are the basic steps in setting up Cyclone Activator. These steps are explained later in more detail.

1. Review Before installing on page 12 for things you should do or know about before installing.

2. If you are upgrading from version 5.0 or later, see Upgrading on page 25 before proceeding with the installation of the new version.

   If you are upgrading from version 4.2.x or earlier, see Migrating version 4.2 trading profiles on page 37 before proceeding.

3. Install Cyclone Activator. See Installation procedure on page 13.

4. Start the server. This creates the database tables for the embedded Derby database. See Start the server on Windows on page 19 or Start the server on UNIX on page 20.

5. Open the user interface in a browser. See Open the user interface on page 22. If this is the first time you have logged on, the getting started page displays. This page provides prompts and links for completing the initial configuration.

6. For security purposes, change the password of the admin user. You also might want to add a new administrative user of your own and assign it to the admin role. You can do this by going to the user and roles area.

7. Specify an SMTP server for sending messages. You can do this by following the link on the getting started page or going to the system management area and clicking **Configure the SMTP server**.

8. Start configuring the application. See Getting started on page 133.

9. While configuring the application you may have to edit system files or add your own custom scripts or code. This is normal in order to use or extend some functionality. When you upgrade later to a newer version of the application, you may want to carry forward the configuration in the files you have changed or added. To simplify this, use the application's site directory to document custom changes. This record keeping is helpful not only for upgrading, but for disaster-recovery planning as well. We also recommend that you keep copies of all files that you add to system directories for custom configurations. This includes such things as modified system configuration files, post-processing scripts and custom Java classes. For more information about how to use the site directory, see Planning for upgrades and disaster recovery on page 54.

   If you do not document your changes to system files, tools are available for identifying changed system files after upgrading. For details see File comparison tools for upgrades on page 34.

Before installing

There are certain things you should do or be aware of before installing Cyclone Activator.

- Make sure you know the location of the license.xml file. The system prompts you for it during installation. You cannot install without it. See About the user license file on page 9. If you do not have this file, contact technical support.

- Make sure the date and time are correct on the computer you are installing the application. This can avoid possible problems later.

- If you have downloaded the software or are not installing from a CD, you must launch the installer from a physical mapped directory. UNC is not supported. If you attempt to install through a UNC connection, the installer displays a message advising you to launch the installer through a conventional directory path (for instance, C:\directory name on Windows).

Installation procedure

Use this procedure to install Cyclone Activator.

Steps

1. Make sure the computer meets the criteria in System requirements on page 1.
2. Review Before installing on page 12.
3. If upgrading from version 5.1 or later, see Upgrading on page 25.

   If upgrading from version 4.2.x or earlier, see Migrating version 4.2 trading profiles on page 37.
4. Follow the procedure for your operating system:

   Install on Windows on page 13

   Install on UNIX on page 14

Note: The installer lets you choose whether to migrate company and partner profile data from Cyclone Activator 4.2.x. If you choose this option, the installer exports profiles from the earlier Cyclone Activator and writes the files to [build number]\profiles\staged of the new application's installation directory tree. See Import all profiles on page 39 for how to import the profiles to the trading engine.

Install on Windows

Use this procedure to install on computers with supported Windows operating systems.

Steps

1. Close any applications that might be running.

2. If using an installation CD, place the disk in the CD drive. When the CD menu appears, click **Install** to launch the installation wizard. If the CD menu does not appear after you insert the CD in the drive, use Windows Explorer and double-click the **install.exe** file in the windows directory on the CD.

   If you downloaded a ZIP file containing the software, unpackage the file properly. If using WinZip, for example, select **Actions > Extract**. Do not unpackage by copying the files from the WinZip window and pasting to a directory, as this corrupts the directory structure of the installation files, causing the installation to fail. After unpackaging the ZIP file, double-click **install.exe** to launch the application installer.

3. Make sure you have a license.xml file. You need this to install the application.

4. Follow the on-screen prompts for installing the application.

   The installer asks whether to set up the server as a Windows service. If you opt for the service, you are prompted to choose your local system account or a domain account. Choose a domain account if the service needs to access your network or other resources requiring user permissions beyond your local account. If unsure, ask your network administrator. Type a domain\user and password only for a domain account; you do not have to do so for a local account. If you decline to set up the Windows service at installation, you can set up the service later. See Run as a Windows service on page 20.

Install on UNIX

Use this procedure to install on computers with supported UNIX operating systems.

The default installation process is text-based, but you can use the **-I gui** parameter to launch a graphical user interface during installation if the computer has X Windows.

The following steps are common to all supported UNIX operating systems. Once you perform them, you are directed to the procedure for your particular operating system to complete the installation.

The home directory for Cyclone Activator must not be automounted or on an automounted file system. Cyclone Activator cannot run correctly on an automounted file system. This applies to volumes mounted using the automount utility and not to volumes that are automatically mounted at startup. Cyclone Activator cannot be installed on automounted volumes because of automatic unmounting of such drives.

Steps

1. Create a user account as the home directory for the application. For example: /opt/cyclone.

2. Check whether there is between 200 and 400 megabytes of storage available in the system /tmp directory. If not, set the environment variable IATEMPDIR to a directory with between 200 and 400 megabytes of disk space. IATEMPDIR is used by the installer program. If required, use the **export** command to change the value of IATEMPDIR from a local variable to an environment variable.

   Alternately, you could clear some or all of the /tmp directory, but doing so could interfere with other processes.

3. Make sure you have a license.xml file. You need this to install the application.

4. If you are installing using a CD, determine the device name of your CD drive. The installation CD has a standard ISO-9660 (High Sierra) file system with Rock Ridge extensions.

   If you downloaded a tar file, copy the file to some directory (for example, cyclone/install). Execute the following command to extract the tar file:

   tar xvf*.tar.

5. See the installation procedure for your operating system:

   Install on Hewlett-Packard HP-UX

   Install on IBM AIX on page 16

   Install on Red Hat Linux on page 17

   Install on Sun Solaris on page 18

Install on Hewlett-Packard HP-UX

Use this procedure to install the application on your Hewlett-Packard HP-UX server with an installation CD.

Steps

1. Log in as **root**.

2. Insert the installation CD into the CD drive.

3 If /mnt does not exist, create it with the following command:

```
mkdir /mnt
```

4 Mount the installation CD with the following command:

```
mount -o cdcase /dev/dsk/* /mnt
```

where /dev/dsk/* is the device name of your CD drive.

5 Log out as **root**.

6 Log in to the cyclone account you created previously.

7 Run the following command:

```
/mnt/hpux/install.bin
```

8 Follow the instructions in the installation process.

9 Log out from the cyclone account.

10 Log in as **root**.

11 Unmount the CD by running the following command:

```
umount /mnt
```

12 Eject the CD from the CD drive.

13 Log out as **root**.

Install on IBM AIX

Use this procedure to install the application on your IBM AIX server with an installation CD.

The Logical Volume Manager (LVM) enables the user to specify the physical sectors of the hard drive, or group of hard drives, to use when creating a volume on the AIX. The sectors closest to the center spindle of the disk generally give the fastest, most efficient, input and output reads and writes. The sectors towards the edge of the disk generally give the slowest input and output results. Once the LVM is used to create and mount a volume on the AIX, Cyclone Activator can be installed into a directory on that mount point just as it can with a non-LVM volume. Cyclone Activator software itself is not aware of whether or not this is an LVM mount point.

Steps

1. Insert the installation CD into the CD drive.
2. Log in as **root**
3. Run the command:

   mount -vcdrfs -r -p /dev/cd? /mnt

   where /dev/cd? is the device name of your CD drive. Possible values are 0 through 9. If you have only one CD drive, its number is probably 0.
4. Log out as **root**.
5. Log in to the cyclone account you created previously.
6. Run the following command:

   /mnt/aix/install.bin
7. Follow the instructions in the installation process.
8. Log out from the cyclone account.
9. Log in as **root**.
10. Unmount the CD by running the following command:

    umount /mnt
11. Eject the CD from the CD drive.
12. Log out as **root**.

Install on Red Hat Linux

Use this procedure to install the application on your Red Hat Linux server with an installation CD.

Steps

1. Insert the installation CD into the CD drive.
2. Log in as **root**
3. Run the command:

mount /dev/cdrom /mnt

where /dev/cdrom is the device name of your CD drive.

4. Log out as **root**.
5. Log in to the cyclone account you created previously.
6. Run the command:

   /mnt/linux/install.bin
7. Follow the instructions in the installation process.
8. Log out from the cyclone account.
9. Log in as **root**.
10. Unmount the CD by running the following command:

    umount /mnt
11. Eject the CD from the CD drive.
12. Log out as **root**.

Install on Sun Solaris

Use this procedure to install the application on your Sun Solaris server with an installation CD. Note that separate procedures are provided, depending on whether you are running the automounter.

Steps

With automounter (the Solaris default)

1. Log in to the cyclone account you created previously.
2. Insert the installation CD into the CD drive.
3. Run the following command:

   /cdrom/cyclone/solaris/install.bin
4. Follow the instructions in the installation process.
5. Eject the CD by running the following command:

```
eject cdrom
```

Without the automounter

1. Insert the installation CD into the CD drive.
2. Log in as **root**.
3. Run the command:

   ```
   mount /dev/sr? /mnt
   ```

   where /dev/sr? is the device name of your CD drive. Possible values are 0 through 9. If you have only one CD drive, its number is probably 0.
4. Log out as **root**.
5. Log in to the cyclone account you created previously.
6. Run the command:

   ```
   /mnt/solaris/install.bin
   ```
7. Follow the instructions in the installation process.
8. Log out from the cyclone account.
9. Log in as **root**.
10. Unmount the CD by running the following command:

    ```
    umount /mnt
    ```
11. Eject the CD from the CD drive.
12. Log out as **root**.

Start the server on Windows

Use one of the following methods to start the server on Windows and you are not using the Windows service.

- Select **Start** > **All Programs** > **Cyclone** > **Start Server**.
- In Windows Explorer, go to [install directory]\[build number]\bin and double-click **startServer.cmd**.

A command window displays as the server is starting. The message **Server Startup Complete** displays when the server has started. Do not close the window or the server stops.

Start the server on UNIX

To start the server on UNIX, log in to the cyclone account you created during the installation process. Run the following command:

[install directory]/[build number]/bin/startServer

Run as a Windows service

You can run Cyclone Activator server as a Windows service. This causes the application to start or stop with the computer. When Cyclone Activator is running as a service, the only visual clues of the server status outside of the user interface are the application's log files and the services area of the Windows computer management window.

When the server has been started as a service, do not try to start it again on the **Start** menu or by double-clicking **startServer.cmd** in the application's bin directory.

The installation wizard asks whether you want to run the server as a service. If you declined, you can install the service now.

If you plan to run Cyclone Activator as a Windows service, we highly recommend creating a user account solely for starting the service upon log-on for Cyclone Activator and related applications (for example, AS2 Accelerator, aXML Agent). This user account should be granted all Windows administrator rights. The account should be fully dedicated to Cyclone Activator and not be used by any other user.

The dedicated user account is recommended because if a user logs on to a Windows computer, whatever applications were started are terminated by Windows when that user logs out.

After installing Cyclone Activator or related application as a Windows service, open the properties for the service and change the default setting of "Local System account" to "This account." Then type the name and password of the dedicated user account. When the Windows computer starts up, the service will start as that special log-in account. So long as no one manually logs on to the Windows computer as that special log-in account name, the application will not be aborted when another user logs out of the computer.

The application installer provides the option to set up the Windows service under a local system account or a domain\user account. Choose a domain account if the service needs to access your network or other resources requiring user permissions beyond your local account. If unsure, ask your network administrator. Type a domain\user and password only for a domain account; you do not have to do so for a local account. If you decline to set up the Windows service at installation, you can set up the service later.

Even if you do not run Cyclone Activator or related applications as a Windows service, a dedicated user account for the Windows computer is still recommended. When the dedicated account is used to start the Windows computer and then Cyclone Activator is started manually, make sure the dedicated user never logs out to ensure the application will run continuously.

Install service

If you are not already running the server as a Windows service and want to, do the following.

Steps

1. Open a command prompt window.
2. Change the directory to [install directory]\[build number]\bin.
3. Run one of the following commands, depending on whether you want the service to run under your local system account or a domain\user account.

   Local account: **cycloneservice -i**

   Domain account: **cycloneservice -i domain\user password**

   Use the domain account option if the service needs to access your network or other resources requiring permissions beyond your local account. If unsure of the option to use, ask your network administrator.

   After running the command, the server will start the next time the computer starts. Or, if the server is not running, you can manually start the service in the current session.

Uninstall service

If you want to stop running the server as a Windows service, do the following.

Steps

1. Open a command prompt window.
2. Change the directory to [install directory]\[build number]\bin.
3. Run the command **cycloneservice -u**.

   The next time the computer starts, you must manually start the server.

Use postInstall to add Windows service

There is another option for installing a Windows service for running Cyclone Activator server. This performs the same action as described in Install service on page 21, but also installs Start menu shortcuts.

Steps

1. Double-click **postInstall.cmd** in [install directory]\[build number]\tools and follow the prompts.

   If a debug message appears upon executing the tool, ignore it.
2. Verify that the shortcuts were created on the Start menu and the service has been added. To check for presence of the service, right-click **My Computer** on the desktop and select **Manage**. On the Computer Management window, expand Services and Applications and click **Services**. The service is named **CycloneService**.

Open the user interface

If you are opening the user interface for the first time, a getting started page displays when you open the UI. This page provides tips for configuring the application.

Before logging on the first time

Before logging on for the first time to the user interface, make sure:

- The server is running. See Start the server on Windows on page 19 or Start the server on UNIX on page 20.
- Internet Explorer 6 or later or Mozilla Firefox 1.0 or later is installed on your computer.

Log on procedure

Use this procedure to log on to the user interface.

Steps

1. Make sure the server is running.
2. When you are ready to log on, use one of the following methods:
   - On Windows, select **Start** > **All Programs** > **Cyclone** > **Admin**.
   - In Windows Explorer, double-click **admin** in [install directory]\[build number]\bin.
   - On Windows or UNIX, point the browser to:

     http://host:6080/ui/

     Host is the fully qualified domain name or IP address of the computer running Cyclone Activator server.
3. Use **admin** as the user ID and **Secret1** as the password when logging on the first time. The user name and password are case sensitive.

   These are the user ID and password of the default system user. After logging on, we recommend creating a user with all permissions enabled and using it as a system administrator. We also recommend immediately changing the password of the user **admin**. See The admin user on page 74..

   The first time you log on a page titled **Getting started** displays. It provides tips and links for configuring the application.

Configuring external SMTP server

When you have started the server and logged on to the user interface the first time, one of the first things you should do is configure an external SMTP server. The system uses this SMTP server by default to send mail, unless its use is overridden in certain cases.

Go to the system management area of the user interface and click **Configure the global external SMTP server** to set up the server.

Consult with your network or e-mail administrator for the information needed to complete the configuration fields.

Stop the server

Use one of the following methods to stop the server:

On Windows, type **stop** in the server console window and press **Enter**.

On UNIX, execute **stopServer**.

Uninstalling

Use this procedure to uninstall Cyclone Activator.

Steps

1. Stop the server.
2. If set up as a Windows service, uninstall the service. See Uninstall service on page 22.
3. Delete the installation directory.
4. If uninstalling on a Windows computer, delete the Start menu shortcuts and any desktop shortcuts.

❖ ❖ ❖

3 Upgrading

When you install Cyclone Activator, BEA WebLogic Edition the top-level directory tree looks like this (for example, on Windows):

```
C:\[install_directory]    \bn
                          \common
```

For the **bn** directory, the **b** stands for build and **n** is a placeholder for the build number, which is a way of representing the software version. The build number directory contains system files. The common directory contains user data and certain files to be retained in an upgrade.

This directory structure is integral to a strategy for seamless upgrades to future versions of this software. When upgrading, the new version is installed in the same root directory (for example, C:\[install_directory]). A new build number directory is created for the new version and resides alongside the build number directory of the old version. The build number directories contain the system files of their respective versions. After upgrading, there remains a single common directory, which contains files shared by both versions. For example, after installing an upgrade, the top-level directory tree looks like this (again, on Windows):

```
C:\[install_directory]    \b1
                          \b2
                          \common
```

We recommend making no changes to these directories as installed. This is not a restriction on the location of integration and backup directories. You can set up such directories inside or outside the application directory tree. If inside the tree, create subdirectories of common. However, we urge against creating subdirectories of the build number directory, as those would not be replicated in an upgrade.

Procedure

- Upgrading from version 5.1 or later on page 26

Concepts

- Changing databases on page 30
- File comparison tools for upgrades on page 34

- Migrating version 4.2 trading profiles on page 37

Upgrading from version 5.1 or later

Use this procedure if you are running version 5.1 or later of Cyclone Activator and intend to upgrade to a newer version.

Note: If you are running a version 5 earlier than 5.1 of Cyclone Activator, contact technical support for help in upgrading. If upgrading from Cyclone Activator 4.2.x, this procedure covers how to migrate profiles from the earlier version to the new version.

You can upgrade from Cyclone Activator 5.1 to a newer version of Activator with complete data integrity.

Cyclone Activator 5.0 through 5.3.1 use Sybase SQL Anywhere as the embedded database. Version 5.3.1.0.1 and later use the Apache Derby database. To migrate data from a Sybase Activator to a Derby Activator, you must use a tool.

If upgrading from a Derby Activator to a newer Derby Activator, the installer handles the data migration.

For information about the data migration tool, see Changing databases on page 30.

If you are unsure of your current database, see Database configuration tool on page 50 to look up database information.

Various upgrading scenarios are possible, depending on whether you want the same or different functionality in the new version. You can continue using the same user license, presuming it has not expired and you want the same functionality after upgrading. You need a new user license if you want to add or change functionality. Technical support can answer questions regarding user licenses.

CAUTION: If you made changes in the previous version that involved editing system files, post-processing scripts, in-line processing, pluggable transports, parsing or other custom changes, you must take steps to carry those modifications forward to maintain the same functionality. Before starting the new server, check the previous version's build number site directory for notes and files about custom changes. Use the notes as a guide for making the same configuration in the new version. If you did not document changes, take inventory of the custom changes in the previous version. Failure to account for custom changes may result in the new version performing below expectations. See Planning for upgrades and disaster recovery on page 54.

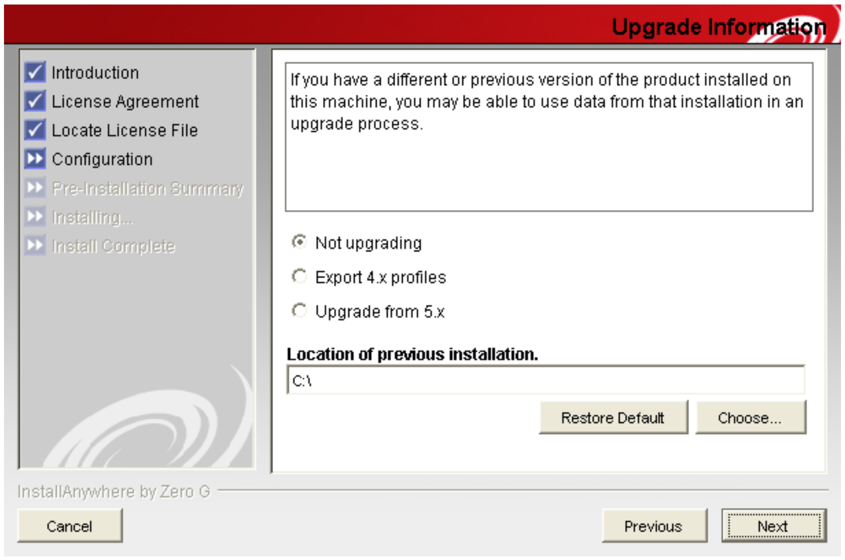
Steps

1 Review System requirements on page 1 to make sure your hardware and software meet requirements for the application upgrade.

2 Stop Cyclone Activator server of the currently installed application.

3 Install the new version, following the instructions for your operating system in Installation procedure on page 13.

The installer prompts for upgrade information. If installing on UNIX, these are text prompts. If installing on Windows, the prompts display on the window in Figure 1.



If you have a different or previous version of the product installed on this machine, you may be able to use data from that installation in an upgrade process.

[✓ Not upgrading] Export 4.x profiles Upgrade from 5.x

Location of previous installation.
C:\

Figure 1. Installer upgrade information window

The following explains the upgrade options.

Not upgrading

Select **not upgrading** when:

Installing for the first time, or

Upgrading from a Sybase Activator to a Derby Activator and you want to migrate data, or

Upgrading from a previous version, but you do not want to migrate data.

Export 4.x profiles

Select **export 4.x profiles** when upgrading from Cyclone Activator 4.2.x and you want to migrate company and partner profiles. You also must specify the path to the 4.2.x installation directory. For more information about this option, see Migrating version 4.2 trading profiles on page 37.

Upgrade from 5.x

Select **upgrade from 5.x** when upgrading from a Derby Activator to a Derby Activator.

Point the installer to the build number directory of the currently installed version. This is an important step. For example, if the current version is installed in C:\[install directory]\[build number], use that path as the location of the previous installation.

Later, regardless of the upgrade option you select, the installer prompts you to choose an installation directory. If upgrading from an earlier version 5, normally you should select the installation directory of the previous version. This places the build number directory of the new version in the same tree with the build number directory of the older version. Figure 2 is an example of this directory structure on Windows.

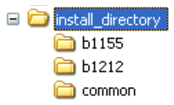


Figure 2. New and old build number directories in same tree

If you choose to install in another directory, copy the contents of the older version's common\conf directory to the new version's common\conf directory. This is necessary so the new version has the same certificate information as the older version.

CAUTION: After installation is completed, do not start the application server until advised later in this procedure.

4 If upgrading from Cyclone Activator with a Sybase database to Cyclone Activator with a Derby database and you want to migrate data to the new Activator, copy the **sybase** directory from the old version to the new version. Specifically, do the following:

Copy this directory:

[install directory]\[**old** build number]\corelib\db\sybase

Paste the directory to:

[install directory]\[**new** build number]\corelib\db

You must do this before using the data migration tool in step 5.

If upgrading from a Derby Activator to a Derby Activator, skip this step.

5 If upgrading from Cyclone Activator with a Sybase database to Cyclone Activator with a Derby database and you want to migrate data to the new Activator, see Changing databases on page 30 for how to use a tool for exporting data from the Sybase Activator.

If upgrading from a Derby Activator to a Derby Activator, skip this step.

If you selected the **Upgrade from 5.x** option instead of the **Not upgrading** option in step 3, check the database connection settings for the newly installed application before using the data migration tool. This is necessary because if you selected **Upgrade from 5.x**, the Sybase database settings were carried forward to the newly installed application and must be changed. See Database configuration tool on page 50 for how to view and change settings. The correct Derby settings are:

Database	Apache Derby
DB host	localhost
Port	0
DB name	Cyclone
User name	dba
Password	sql

6. If you have changed any system files or added custom scripts or code in the older version, refer to the site directory for your notes and copies of custom files so you can carry the same functionality forward to the newly installed application. See Planning for upgrades and disaster recovery on page 54.

   Alternately, tools are available to help you identify system files that have been changed since the previous version was installed. Use of these tools is explained in File comparison tools for upgrades on page 34.

   Once identified, you can locate your changes in the older system files and make similar changes in the matching system files for the newly installed application. Do not replace newer versions of system files with older versions.

7. If you exported database files in step 5 to change databases, use the data migration tool to import the data. See Changing databases on page 30.

8. Start the server.

9. Check the message validation rules for community profiles to make sure the rules are correct for signing and encryption of inbound messages, if upgrading from a version that did not have such rules (versions earlier than 5.3.2).

10. When you are satisfied the new version is operating properly, you can delete, or backup and delete, the build number directory for the old version at your discretion. You might want to retain the directory if there is a possibility of reverting to the old version. If you do retain the old version directory, do not try to run the old and new versions at the same time on the same computer or with the same database.

Changing databases

Cyclone Activator 5.0 through 5.3.1 use Sybase SQL Anywhere as the embedded database. Version 5.3.1.0.1 and later use the Apache Derby database. To migrate data from a Sybase Activator to a Derby Activator, you must use a tool.

If you do not want to migrate data when upgrading, skip this topic.

The tool is Data Mover in [install directory]\[buld number]\tools. Using the tool requires configuring the datamover.properties file in [install directory]\[buld number]\conf.

Note that the tool can be used only with version 5 or later. It cannot be used with version 4.2.x and earlier. Nor can it be used to import to version 5 or later a database file exported by version 4.2.x or earlier.

Configure properties file

Running the Data Mover tool from a command line is a two-step operation for both exporting and importing. Before exporting, you must edit a properties file and then run the tool. Before importing, you must once more edit the properties file and then run the tool again.

The following describes the properties in the datamover.properties file that control the behavior of the Data Mover tool. The file is in [install directory]\[buld number]\conf.

If you need information about your current database, use the database configuration tool before editing datamover.properties. The database configuration tool provides information such as database type, database host name, port type, database name and the user name for connecting to the database. See Database configuration tool on page 50.

We recommend making a backup copy of the original properties file before editing in case you need to go back to the first file.

datamover.WorkingDir

If exporting, the path to the directory where the database files are written. If exporting a large database, make sure the target directory has enough available space for the files.

If importing, the path for the database files to import.

This can be a relative or absolute path. If the directory does not exist, the tool creates it upon exporting data.

datamover.Operation

The action to perform. Values are:

export
import

datamover.ResetCluster

If exporting, set this property to **false**.

If importing, set to **false**.

The following is an exception when importing. When importing and moving from a single VM to a multiple VM version of the application (for example, from Activator to Cyclone Interchange AS), set to **true**.

datamover.Db.Type

If exporting, the source database type.

If importing, the target database type.

Values are:

Derby
Sybase

datamover.Db.Host

If exporting, the name of the computer running the source database. If exporting from a Sybase Activator, the default is **localhost**.

If importing, the name of the computer running the target database. If importing to a Derby Activator, the default is **localhost**.

datamover.Db.Port

If exporting, the port on which the source database listens.

If importing, the port on which the target database listens.

Default ports are:

Derby	0
Sybase	2639

datamover.Db.DatabaseName

If exporting, the name of the source database. If exporting from a Sybase Activator, the default name is **Cyclone**.

If importing, the name of the target database. If importing to a Derby Activator, the default name is **Cyclone**.

datamover.Db.UserName

If exporting, the user name for connecting to the source database. If exporting from a Sybase Activator, the default name is **dba**.

If importing, the user name for connecting to the target database. If importing to a Derby Activator, the default name is **dba**.

datamover.Db.Password

If exporting, the password for connecting to the source database. If exporting from a Sybase Activator, the default password is **sql**.

If importing, the password for connecting to the target database If importing to a Derby Activator, the default password is **sql**.

Running Data Mover

After configuring the datamover.properties file to export or import data, run the command-line script **dataMover.cmd** (Windows) or **dataMover** (UNIX). The script is in [install directory]\[build number]\tools. Run it from the tools directory.

Make sure to stop Cyclone Activator server before executing an export or import action.

CAUTION: If upgrading from Sybase Activator to Derby Activator, use Data Mover in the order recommended in the procedure Upgrading from version 5.1 or later on page 26.

To run the tool, you only need to type the script name and execute. No parameters are required. The tool gets the information it needs from the properties file.

Before using the tool, see the Sybase notes and Derby notes.

Sybase notes

If you are exporting from a Sybase database, make sure the sybase.properties file in the conf directory of the newly installed application correctly points to the Sybase database you want to export. The two settings that need to be correct are bin_path and database_file.

For example, in the newly installed application these properties are:

```
bin_path=../corelib/db/sybase
database_file=../corelib/db/sybase/Cyclone.db
```

If you are exporting the Sybase database from the previous version in the b1172 directory, edit the properties as follows:

```
bin_path=../b1172/corelib/db/sybase
database_file=../b1172/corelib/db/sybase/Cyclone.db
```

Derby notes

If you are exporting to or importing from Derby, the tool assumes the Derby database is in the ../corelib/db/derby directory relative from where the application is run. To change the location of the Derby database, edit the Dderby.system.home property in the dataMover script.

The console window displays the progress of the export or import action. Importing may take several minutes longer than exporting. There also is a log file named DataMover.log in the logs directory.

When exporting, the tool writes hundreds of files to the working directory specified in the properties files. The number of files can range from 250 to 350 or more.

The files must be imported to a fresh, never-used database. To ensure this, install Cyclone Activator, but do not start the application server before importing the database files with Data Mover.

CAUTION: The database files should be imported to the new database before Cyclone Activator server is ever started. After importing the data, the server can be started for the first time.

After importing the data and starting the server, you can delete the working directory for the data files.

File comparison tools for upgrades

If you have changed any system files in the build number directory, we recommend checking whether you need to carry over custom settings when upgrading to a new version. For example, you might have edited the events.xml, log4j.properties or other configuration files and want the same settings after upgrading. Although database records are preserved in an upgrade, customized system files are not carried over from the old to new version.

Three command-line tools are available to help you identify system files that might have been changed after the previous version was installed or that are different between the old and new version. The tools are in [install directory]\[build number]\tools of the new version. All can be invoked with the -? parameter to generate syntax usage help.

The tools useful for you depend on the version you are upgrading from. The following explains the tools.

upgradeList

When installing 5.2, upgradeList is called by the installation process and generates a snapshot of the entire application installation directory tree. After installing, you can run the upgradeCompare tool and compare saved snapshots.

You can run this tool by itself without parameters or use the -? parameter to display help text. The tool writes to a file named cycloneInstallation.txt. Note that if you run upgradeList without parameters, the original snapshot file created at installation time is overwritten. The following is a sample of the content.

```
\ 1093367237950 0
\CycloneInterchangeREADME.txt 1089734570000 34680
\JettyLicense.pdf 1093337636000 20341
\TransactionDirectorREADME.txt 1088705501000 22248
\bin\ 1093367241064 0
\bin\CycloneService.exe 1090275374000 94208
\bin\CycloneService.ini 1091833623000 1373
\bin\OBOE.properties 1068851942000 566
\bin\host_environment.cmd 1093367239943 700
\bin\ViewExecutiveLog.cmd 1086886108000 141
\bin\admin.url 1093367239422 97
\bin\as1Tool.cmd 1076460741000 513
\bin\as2Tool.cmd 1076460740000 513
\bin\as3Tool.cmd 1076460857000 513
\bin\asxTool.cmd 1076460899000 513
\bin\createMachineEnv.cmd 1090695608000 635
\bin\db2_create.sql 1089754151000 2743
\bin\db2_runstats.cmd 1074123898000 309
\bin\db2_runstats.sql 1074123896000 414
\bin\dbConfig.cmd 1086372490000 895
\bin\dirTester.cmd 1089670945000 381
\bin\environment.cmd 1093367239402 1116
\bin\ftpTester.cmd 1090948658000 625
\bin\keyInfoWriter.cmd 1090865832000 1139
\bin\postInstall.cmd 1093367239422 89
\bin\postInstall.dat 1093337859000 1483340
\bin\startServer.cmd 1087248742000 1314
```

upgradeDiff

The upgradeDiff tool recursively compares the sizes of system files in the old and new installation directory trees. The focus of the comparison are the bin, conf and corelib directories and their subdirectories. These are the most likely directories where changed system files can be found. You can, however, alter the search to include or exclude other directories.

This tool is helpful if you are upgrading from a pre-5.2 version and do not have a directory snapshot of the old tree, such as the upgradeList tool provides for version 5.2 and later.

You can run this tool by itself without parameters or use the -? parameter to display help text. The tool writes to a file named cycloneDifferences.txt. The following is a sample of the content. Lines are truncated. Records with the prefix <Diff> are the files detected to be of different sizes.

```
<Same> \conf\cyclone_event.xsd cyclone_event.xsd is a file that
<Diff> \conf\datastoreconfig.xml datastoreconfig.xml is a file t
<Both> \conf\db\ db is a folder that exists both in C:\cyclone\b
<Both> \conf\db\shared\ shared is a folder that exists both in C
<Same> \conf\db\shared\alerts_schema.xml alerts_schema.xml is a
<Same> \conf\db\shared\cachet_schema.xml cachet_schema.xml is a
<Same> \conf\db\shared\clusterAdmin_schema.xml clusterAdmin_sche
<Same> \conf\db\shared\clusterbus_schema.xml clusterbus_schema.x
<Same> \conf\db\shared\common_schema.xml common_schema.xml is a
<Same> \conf\db\shared\events2_schema.xml events2_schema.xml is
<Same> \conf\db\shared\pcclasses.lst pcclasses.lst is a file th
<Same> \conf\db\shared\persistence_schema.xml persistence_schem
<Same> \conf\db\shared\valence_schema.xml valence_schema.xml is
<Both> \conf\db\submit2tx\ submit2tx is a folder that exists bo
<Same> \conf\db\submit2tx\submit2tx_schema.xml submit2tx_schema
<Both> \conf\db\tradingEngine\ tradingEngine is a folder that e
<Same> \conf\db\tradingEngine\collaboration_schema.xml collabor
<Same> \conf\db\tradingEngine\csos_schema.xml csos_schema.xml i
<Same> \conf\db\tradingEngine\ms_sql_server_procs.xml ms_sql_se
<Same> \conf\db\tradingEngine\oracle_procs.xml oracle_procs.xml
<Same> \conf\db\tradingEngine\sybase_procs.xml sybase_procs.xml
<Diff> \conf\db\tradingEngine\tradingengine_schema.xml tradinge
<Same> \conf\db\tradingEngine\webservices_schema.xml webservice
```

upgradeCompare

The upgradeCompare tool searches for and lists changes made to an installation tree after a snapshot was taken with the upgradeList tool. The upgradeCompare tool compares the installation tree to a snapshot file in the bin directory of the current or previous version.

The tool attempts to analyze the comparison with a prefix for each line in the output file such as <Same>, <Only-Prev>, <Diff-PrevNewer>, <Diff-CurBigger>.

The tool by default searches all directories and subdirectories, but directories can be excluded to narrow the comparison.

You must run upgradeCompare with the **-usecurrenttree** or **-useprevioustree** parameter to indicate whether to compare the snapshot file to the current or previous build number directory. You also can use the -? parameter to display help text. The tool writes to a file named cycloneComparison.txt. The following is a sample of the content. Lines are truncated.

```
<Only> \conf\businessprotocols\asx\ asx is a folder that
<Diff-CurBigger-CurNewer> \conf\businessprotocols\asx\con
<Only> \conf\businessprotocols\ebxml\ ebxml is a folder t
<Diff-CurBigger-CurNewer> \conf\businessprotocols\ebxml\c
<Only> \conf\businessprotocols\email\ email is a folder t
<Diff-CurBigger-CurNewer> \conf\businessprotocols\email\c
<Only> \conf\businessprotocols\mmd\ mmd is a folder that e
<Diff-CurBigger-CurNewer> \conf\businessprotocols\mmd\conf
<Only> \conf\businessprotocols\raw\ raw is a folder that e
<Diff-CurBigger-CurNewer> \conf\businessprotocols\raw\conf
<Diff-CurBigger-CurNewer> \conf\crossworks.properties cross
<Diff-CurBigger-CurNewer> \conf\cyclone_event.xsd cyclone_
<Diff-CurBigger-CurNewer> \conf\datastoreconfig.xml datast
<Only> \conf\db\ db is a folder that exists only in C:\cyc
<Only> \conf\db\shared\ shared is a folder that exists onl
<Diff-CurBigger-CurNewer> \conf\db\shared\alerts_schema.xm
```

Migrating version 4.2 trading profiles

If you are upgrading from Cyclone Activator 4.2.x to 5.0 or later, you can copy company and partner profiles to the new application. You cannot, however, migrate Tracker data.

During installation of the trading engine, the system asked whether you wanted to copy profiles from an earlier version. If you accepted this option, the installer exported the profiles from the older application and wrote the files to [install directory]\[build number]\profiles\staged. See Import all profiles on page 39 for how to import those profiles to the new application.

If you did not export profiles during installation or need to export many profiles again from an earlier version, see Export all profiles for how to use a tool to export all profiles.

If you plan on installing version 5.0 or later on a different computer, you can install the new version on the machine running 4.2.x simply to use the installer's tool for extracting the profiles from 4.2.x. Then copy the exported profile files to the computer you plan to install the new version, and delete the new version from the 4.2.x computer.

Export all profiles

Use this procedure to export all company or partner profiles at the same time from Cyclone Activator 4.2.x. This procedure does manually what the version 5.0 or later installer does automatically.

Although you can use a password to protect the personal certificates exported with this method, you should delete the profiles after you import them to the trading engine, as anyone with a copy of the trading engine software could also import them if the password became known.

Steps

If you used the profile export option when installing the trading engine, that process created the profile tool described in step 1 through step 5. If the tool already is in the Cyclone Activator 4.2.x bin directory, go to step 6.

1. Locate the import utility in the Cyclone Activator 4.2.x bin directory. This file is named **Import.bat** in Windows and **import** in UNIX.
2. Make a copy of the file and name it **profileImportExport.bat** (Windows) or **profileimportexport** (UNIX). Leave the file in the bin directory.
3. Open the file for editing and find the following string at the end of the file:

   upgrade.StartImport
4. Delete the string and type the following string in its place:

   Windows: util.ProfileImportExport %*

   UNIX: util.ProfileImportExport $*
5. Save and close the file.
6. Open a command window to run the utility. You need to run it twice, first to export partner profiles and then company profiles. The order does not matter. Run the utility from the bin directory.

   The following is the syntax for exporting partner profiles:

   profileimportexport -export -partner -all [destination directory]

   The following is the syntax for exporting company profiles:

profileimportexport -export -company -all [destination directory] [password]

Specifying a password is optional for exporting company profiles, but recommended to protect certificate private keys. Remember the password when importing the profiles later.

Use separate destination directories for the partner and company profile files, as this simplifies importing the files later. If a destination directory does not exist, the utility creates it.

7 Go to Import all profiles.

Import all profiles

Use this procedure to import many company or partner profiles at the same time. This procedure typically is used when migrating profile data from Cyclone Activator 4.2.x to 5.0 or later. You must import company profiles before importing partner profiles.

This is not the procedure to use when importing one profile file at a time. For that procedure see Exporting and importing profiles on page 150.

Note: In Cyclone Activator 5.0 and later, company profiles are called community profiles.

Copy profile files to [install directory]\[build number]\profiles\autoImport. Once the server is stared, the trading engine on its own imports any compatible profiles it retrieves from this directory. If the system is unable to import a profile, it moves the file to \profiles\autoImport\rejected. Once imported, the system moves the profile files to the appropriate \profiles\backup subdirectory, either community or partner.

Post-import tasks

After you import profiles, you should check the following items:

- Security settings
- Integration delivery exchanges
- Trading delivery exchanges
- Secondary IDs

Security settings

In Cyclone Activator version 4.x, security settings are specified on a per-partner basis only. In this version, security settings are set up in the collaboration settings and message validation sections of the community profile, where you can set up message security on a community or per-partner basis. Because each version handles these settings differently, the import process cannot translate them properly. You should verify that the partner security settings you used in version 4.x are reflected in this application. See Collaboration settings on page 367 and Inbound message validation on page 399.

Integration delivery exchanges

The import process creates one integration delivery exchange for a community. By default, the target location for the integration delivery uses the following convention:

c:\[previous version's installation directory]\[community routing ID]\in

You should verify that the target location for your integration delivery exchange is suitable for your installation. If you run the trading engine in a cluster, the target location should be a directory on a shared network drive.

You can only specify message routing based on MIME type and binary message handling by calling a post-processing script from the integration delivery exchange. See Manage file system integration on page 227 for more information.

If you change the target location from that used by Cyclone Activator version 4.x, you must also update any back-end systems to retrieve messages from the new locations.

Trading delivery exchanges

If you are migrating XML profiles from Cyclone Activator 4.1.x or earlier, those versions do not export all profile information that Cyclone Activator 5.0 and later requires. Upon importing profiles from version 4.1.x or earlier, check the configuration of trading delivery exchanges. You might have to configure the exchanges after importing the profiles.

Secondary IDs

Secondary IDs in 4.2.x partner profiles import as additional routing IDs in 5.0 and later partner profiles. This might require you to perform additional configuration if secondary IDs were used for re-routing messages in 4.2.x.

❖ ❖ ❖

4 Tools and options

The following topics describe tools available for checking configurations and troubleshooting. Information also is provided about the application's installation directory tree, how to plan for upgrading and disaster recovery, and an overview of the system's single sign-on capability.

Concepts

- Tools in bin directory
- Tools in tools directory on page 44
- Installation directory tree on page 47
- Single sign-on interface on page 52
- Planning for upgrades and disaster recovery on page 54
- Running the UI over HTTPS on page 55

Procedure

- Database configuration tool on page 50
- Binding network interfaces on page 51

Tools in bin directory

The application's bin directory at [install directory]\[build number]\bin contains a number of tools, some of which can be useful in certain cases. The following table summarizes the tools and provides links for more information for those you might have occasion to use.

If you use Windows, the names of these tools have an extension of .cmd (for example, dbConfig.cmd). If you use UNIX, there is no extension (for example, dbConfig).

Depending on whether you use Windows or UNIX, some of these tools may not have been installed with the application. If a tool changes a value in the database, restart the server for the change to take effect.

Table 2 - Tools in the application's bin directory

Tool	Description
[host]_environment	Sets the short and long names of the computer running Cyclone Activator server. This tool is not for use by end users.

Table 2 - Tools in the application's bin directory

Tool	Description
admin	Opens the user interface log-on page in a browser.
dbConfig	Displays and allows you to change connection settings for an external database. See Database configuration tool on page 50.
environment	Sets environment variables that are specific to all nodes running on a particular computer. This tool is not for use by end users.
netConfig	The system uses this utility to identify the short and long names of computers running the trading engine server. On Windows, this tool also manages the contents of the CycloneService.ini file, as well as the <machine>_environment.cmd script that is used when the trading engine is run from a command prompt.
startServer	Starts Cyclone Activator server application. See Start the server on Windows on page 19 or Start the server on UNIX on page 20.
stopServer	The UNIX command to stop Cyclone Activator server application. See Stop the server on page 24
systemuser	Changes or adds a system user and password. This user is only used by Cyclone Activator system. This tool is not for use by end users, except in connection with SSO.
unlockuser	Unlocks a user who is blocked from logging on to the user interface after exhausting the number of log on retries. See Unlocking a blocked user on page 78.

Tools in tools directory

The application's tools directory at [install directory]\[build number]\tools contains a number of tools, some of which can be useful in certain cases. The following table summarizes the tools and provides links for more information for those you might have occasion to use.

If you use Windows, the names of these tools have an extension of .cmd (for example, as1Tool.cmd). If you use UNIX, there is no extension (for example, as1Tool). This does not apply to scripts with extensions of .sql.

Depending on whether you use Windows or UNIX, some of these tools may not have been installed with the application. If a tool changes a value in the database, restart the server for the change to take effect.

Table 3 - Tools in the application's tools directory

Tool	Description
as1Tool	Packages, unpackages and dumps EDIINT messages. This tool is not for use by end users.
as2Tool	Packages, unpackages and dumps EDIINT messages. This tool is not for use by end users.
as3Tool	Packages, unpackages and dumps EDIINT messages. This tool is not for use by end users.
asxTool	Packages, unpackages and dumps EDIINT messages. This tool is not for use by end users.
dataMover	Migrates data from one database to another. This tool is to be used only by some Cyclone Activator users under the supervision of technical support. See Changing databases on page 30.
db2_create.sql	Used in configuring a DB2 database.
db2_runstats	Updates new DB2 database tables and optimizes their performance.
db2_runstats.sql	See db2_runstats.
DeleteUploaded DirectorDocs.sql	This script deletes all records in the Transaction Director database. This tool is for use only upon advice of technical support.
derby_IJ	Enables SQL queries of a Derby database. This tool is not for use by end users.
diagnose	When performing troubleshooting, this tool can be used to compress and send log files to technical support. Technical support often requests log files when helping users. Run the tool from a command line and follow the menu prompts.

Table 3 - Tools in the application's tools directory

Tool	Description
dirTester	Tests the Java temp or other specified directory by writing an unbuffered and a buffered temp file. This tool is for use only upon advice of technical support.
exportProfile	Exports community and partner profiles to XML files. Community profiles are exported as partner profiles. The advantage of this tool is the exporting of partner profiles as partner profiles, an option not available in the user interface. Run exportProfile without parameters to display directions for using the tool. This tool is only for use with Cyclone Activator 5.4 or later.
ftpTester	Verifies interoperability of the trading engine with FTP servers. See FTP tester tool on page 429.
httpTester	Tests whether an HTTP client can connect to the HTTP server. This tool is for use only upon advice of technical support.
jmsTester	Checks for proper configuration of JMS queues. See JMS transport on page 212.
keyInfoWriter	Extracts KeyInfo element information from a certificate for use in a CPA for ebXML trading. See Extracting the KeyInfo element for a CPA on page 469.
logViewer	Interleaves multiple log files and sorts log entries chronologically. It also can filter log categories, log levels and threads. This tool is not for use by end users.
messagePurgeTool	Immediately deletes all database records of traded messages and all files in the backup directory. See Purge all records, backup files on page 424.
netInfo	Finds network interfaces for a computer. See Binding network interfaces on page 51.
peerNetwork	

Table 3 - Tools in the application's tools directory

Tool	Description
postInstall	Sets up Cyclone Activator server as a Windows service and adds shortcuts on the Start menu. See Use postInstall to add Windows service on page 22.
postInstall.dat	See postInstall.
rejectInprocess Messages	Sets Cyclone Activator messages that are stuck in the in-process state to a status of failed. This tool is for use only upon advice of technical support. For use only when nodes are stopped.
sftpTester	Verifies the operation of the SFTP client in the trading engine and a partner's SFTP server. See SFTP transport on page 206.
update	Updates the database. This tool can be used only when instructed after receiving an update.dat file from technical support.
upgradeCompare	Searches for and lists changes made to an installation tree after a snapshot was taken with the upgradeList tool. This tool is used when upgrading. See File comparison tools for upgrades on page 34.
upgradeDiff	Recursively compares the sizes of system files in the old and new installation directory trees. This tool is used when upgrading. See File comparison tools for upgrades on page 34.
upgradeList	Generates a snapshot of the entire application installation directory tree. This tool is used when upgrading. See File comparison tools for upgrades on page 34.

Installation directory tree

Application files used by Cyclone Activator are in two primary directories under the installation directory. These are the build number directory and the common directory.

The build number directory holds application files. The name of the build number directory is in the format **bn**, where **n** is a number. A build number is another way of representing the version of the software. The subdirectories and files in the build number directory are specific to a particular version.

The common directory holds files used by a particular installation of Cyclone Activator. These files are not specific to a particular software version and could be re-used when a newer version is installed later.

There is a third directory at the same level as the build number and common directories. It is the JRE directory that holds the Java Runtime Environment files used by the application. This directory is named in the format **jre_n.n**, where **n.n** stands for the JRE version number. In versions before 5.4, the JRE directory was under the build number directory.

The following topics describe the subdirectories of the build number and common directories. These are general descriptions to provide a high-level familiarity and not a detailed accounting of all subdirectories and files.

[install directory]\[build number]

The following table describes the subdirectories of the build number directory.

Table 4 - Build number directory descriptions

Directory	Description
bin	Holds scripts or tools for starting the server, installing Windows service, changing database connection settings. Also contains environment files.
conf	Stores configuration files for alerts, events and many other functions. This directory also is the home of the license.xml file, which controls application functionality permissions.
corelib	Repository for JARs of third-party applications used by Cyclone Activator. This directory also stores database drivers.
doc	User documentation in PDF format for Cyclone Activator. This is in addition to a help system that is accessible on the Help menu in the user interface.
image	Contains application icon files used on the Start menu on computers with Windows operating systems.
jars	Repository for JARs proprietary to Cyclone Activator.
logs	Holds most of the log files generated by the system.
profiles	This is where XML profile files can be copied for manual or automatic importing to Cyclone Activator.

Table 4 - Build number directory descriptions

Directory	Description
samples	Contains sample code for demonstrating uses of various optional features.
site	Recommended directory for users to document custom changes made to Cyclone Activator and for storing custom scripts and code as an aid for upgrading or disaster recovery.
tools	Has scripts for executing various troubleshooting and configuration tools.
util	Contains files and user documentation for optional utilities for Cyclone Activator and Transaction Director.
webapps	Stores system files used by web services and the user interface.

[install directory]\common

The default location for the common directory is [install directory]\common. In a clustered environment, the common directory must be shared so it is accessible by all nodes. Usually this means the common directory is on a network file system.

The location of the common directory is determined during installation. The commonPath entry in the filereg.xml file notes the location of the common directory. The filereg.xml file is in [install directory]\[build number]\conf

The following table describes subdirectories of the common directory. Not all of these directories may appear or be used on your file system.

Table 5 - Common directory descriptions

Directory	Description
conf	Stores security files such as certificate keys and certificate revocation lists.
cpatemplates	Where Cyclone Activator stores imported CPA templates used for ebXML trading.
data	Contains copies of files processed by Cyclone Activator. Message tracker uses these files. Users can control the volume of files through controls in the user interface.

Table 5 - Common directory descriptions

Directory	Description
diagnose	This directory is used by the diagnose log file packaging tool to store compressed files. See diagnose on page 45.
filestore	Contains copies of the files uploaded to Transaction Director.
logs	Transaction Director writes a log file to this directory when GENERATE_AUDIT_LOGS=true in the txserver.properties file. By default this log is turned off.
webtrader	Cyclone Activator default directory for storing documents exchanged between a web trader and a community sponsor.

Database configuration tool

Database driver and connection information for Cyclone Activator is in a file named datastoreconfig.xml in [install directory]\[build number]\conf. A tool is available for viewing information in this file.

Use the database configuration tool only when the server is not running. When running, the database port is in use and the tool cannot connect to the database.

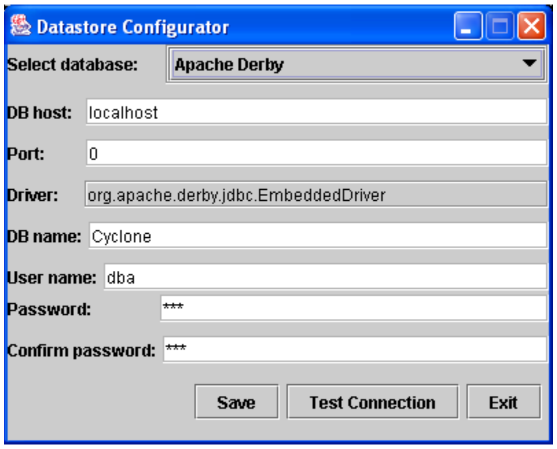
CAUTION: Users of Activator must not use this tool to change database settings. Only use the tool if you want to review or test database settings. Also, do not manually change any database settings in the datastoreconfig.xml file.

The database configuration tool is named dbConfig.cmd for Windows and dbConfig for UNIX. It is in [install directory]\[build number]\bin.

The tool can be used with a graphical user interface or executed from a command line.

To run the tool in GUI mode, in Windows double-click **dbConfig.cmd** or run **dbConfig.cmd** in a command window. In UNIX run **dbConfig** in a console window.

Figure 3 shows the database configuration window. The fields display the current database information in datastoreconfig.xml.


Datastore Configurator

Select database: Apache Derby

DB host: localhost

Port: 0

Driver: org.apache.derby.jdbc.EmbeddedDriver

DB name: Cyclone

User name: dba

Password: ***

Confirm password: ***

Figure 3. Database configuration window

The tool can be run in a command line with the following parameters:

[-? | -help] [-d (sybase|sqlsrvr|ora|db2|derby)] [-p port] [-h host]
[-n dbname] [-u username] [-pd password]

When you execute the tool from a command line, you can change information, but not display current information as with the GUI.

Here are some guidelines for using the database configuration tool.

1. Before using the tool, close Cyclone Activator server.
2. If you change the database type to Oracle, copy ojdbc14.jar to [install directory]\[build number]\corelib\db\oracle.
3. To use the test connection feature in the GUI, change the fields you want and click **Test Connection**. If the database has been created, the information in the fields is correct and the tool can connect to the database, the test should succeed. If not, make changes and test again. When you are satisfied with the results, click **Save**.
4. When using the GUI, click **Save** before exiting or your changes are not saved. Changes are saved to datastoreconfig.xml in the application’s conf directory.
5. Restart Cyclone Activator server when you are done.

Binding network interfaces

This topic describes an advanced control you can ignore unless binding of servers is important from a network management perspective.

By default, all of this application's internal socket servers (for example, HTTP, SMTP) are bound to all network interfaces of the computer used as the application server, or computers used as servers if a clustered environment. This allows clients to connect to the servers on any of the IP addresses associated with the network interfaces in a given machine. For example, if a machine has two network interfaces having IP addresses 10.10.1.1 and 10.10.1.2, the HTTP server for the user interface would be available on both IP addresses at the default port of 6080. The same would apply to the HTTP server for trading (if configured), but at the default port of 4080. Note that all servers also would be available on the loop-back interface, 127.0.0.1, as well (if available). For the majority of users, the default behavior of binding to all available IP addresses is satisfactory.

The other option is to bind only to the IP address of the application server identified by the CYC_NETWORK_NAME property. This property is in the [machine_name]_environment file, or files if a clustered environment. The file is in [install directory]\[build number]\bin.

To bind only to CYC_NETWORK_NAME, go to the system management page in the user interface and click **Configure system properties**. Select the CYC_NETWORK_NAME option and click **Save changes**. Restart the server for the change to take effect.

There is a command-line tool in the application's tools directory named **netInfo** that you can use to identify the network interfaces on the application server. Note that it may report more than one IP address per network interface, depending on how your network is configured. The tool also reports the value of CYC_NETWORK_NAME. Run the script without parameters from the tools directory.

Single sign-on interface

Cyclone Activator has a pluggable interface that allows connection to an external single sign-on (SSO) system. This interface supports the product's existing Cachet SSO interface, but can be replaced by a user implementation that accesses an SSO repository.

This pluggable interface actually has multiple interfaces that can be implemented separately, depending on the required level of SSO integration. The interfaces are:

UserAuthority	Verifies users and permissions.
UserManager	Creates or modifies users or groups.
Notification	For use by a remote system to notify Cyclone Activator server when a user is invalid.

Cyclone Activator internally uses the same SSO architecture that is available to an external SSO system. Figure 4 shows a simple comparison of external and internal implementations.

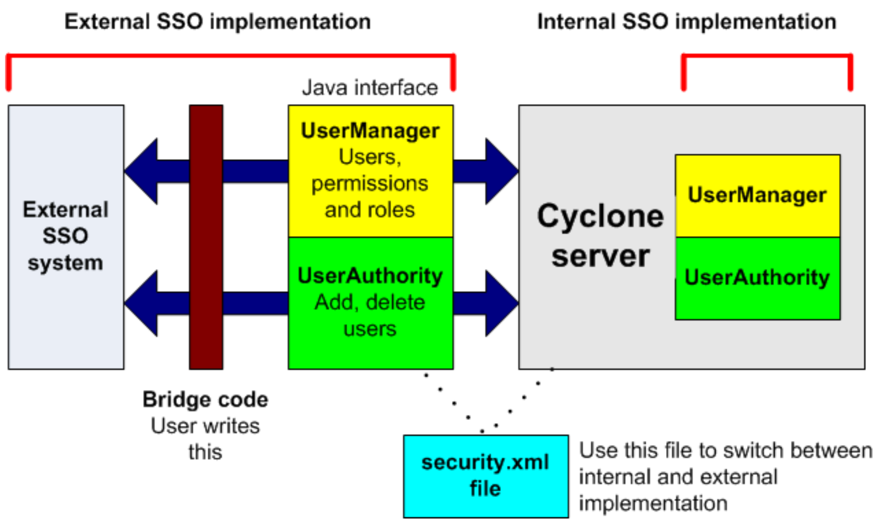


Figure 4. SSO external and internal implementation

Documentation and sample code are provided to help you implement the SSO framework for an external SSO system. The following describes the documentation and sample code and where you can find it. The referenced directories are subdirectories of [install directory]\[build number].

Documentation

The following SSO documentation is provided.

SsoDemoReadme

SsoDemoReadme is a text file with instructions for building a demonstration SSO implementation. This is for demonstration purposes only and should not be used in production.

Location: \samples\sso

User Authority and User Management Guides

This documentation describes the SSO interfaces and the functionality of user management, user authority and cache management.

Location: \samples\sso\docs

Double-click **index.html** to open the documentation in a browser.

Javadoc

The Javadoc describes the Java classes for SSO.

Location: \samples\sso\docs\javadocs

Double-click **index.html** to open the Javadoc in a browser.

Sample code

Sample code is available to create a demonstration SSO implementation. The SsoDemoReadme file explains how.

Location: \samples\sso\src\demo

Planning for upgrades and disaster recovery

Besides following your company's policies for backing up data and applications, steps are recommended to make upgrading and disaster recovery easier for Cyclone Activator.

Managing Cyclone Activator typically involves making custom changes to fit your processing needs. This can mean editing system files or adding your own scripts and Java classes for purposes such as post-processing, in-line processing, parsing, pluggable transports or other custom changes.

The extent of custom changes depends on the complexity of your configuration. Regardless of the complexity, the best practice is to document your changes and keep copies of custom scripts or code files.

Cyclone Activator provides a file system directory to help in documenting custom changes. The directory is [install diretory]\[build number]\site. A readme text file there provides an overview. An advantage of using the site directory is that upon upgrading to a new version, the contents of the site directory from the old installation directory tree are copied to the new version as part of the installation process.

The site directory has subdirectories with the following recommended uses.

bin

Store copies of your post-processing scripts, other scripts or executable files.

Where possible in the user interface, use a relative path to point to this directory (for example, for a post-processing script). After upgrading to a new version, you will not have to alter the path in the UI.

conf

Store copies of custom configuration files.

doc

Store text documents containing notes about custom changes. These can be notes about any changes that would be a useful reference for someone peforming an upgrade or disaster recovery.

For example, if you edit the alerts.xml or events.xml file in [install directory]\[build number]\conf, document the changes in a text file and save it here. When upgrading, use the notes to make the same changes to the alerts.xml or events.xml in the new version.

Note that it is not recommended to make backup copies of changed system files for the purpose of substituting the backed up files for ones in a newly installed application. This is because system files may have been changed by the software developers between an old and new version of the application. This is especially true of the filereg.xml file. The filereg.xml file installed with a new version should always be used.

Custom changes for some values in system files do not require documenting because they are forwarded during an upgrade by the application installer. This includes the commonPath entry from filereg.xml, the certs and crls paths in crossworks.properties, and the filestore and audit log paths in txserver.properties.

jars

Store Java archive files for in-line processors, pluggable transports, JMS, custom parsers. Your custom code should reference this directory as ../site/conf. Any JAR copied to this directory automatically is included in the classpath before [install directory]\[build number]\jars.

Running the UI over HTTPS

The default way for browsers to connect to the application server's user interface is via HTTP. The typical URL used by a browser is: http://hostname:6080/ui. Optionally, you can have browsers connect via HTTPS (HTTP over SSL). The following topics explain how to configure this.

Use keytool

Configuring browsers to connect via HTTPS requires using **keytool**, a security tool in the Java Development Kit. For documentation explaining the use of keytool, go to:

http://java.sun.com/j2se/1.5.0/docs/tooldocs/index.html#security

Self-signed or CA certificate

The server can use a self-signed certificate or one issued by a certificate authority. Upon connecting, the server presents the certificate and public key to the browser to establish the secure connection.

If you have a CA-issued signed certificate, see the keytool documentation and the instructions from your CA for ways to create your keystore. Otherwise, you can use keytool to create your own self-signed certificate. The following is an example of a keytool command for creating a self-signed certificate:

```
    keytool -genkey -alias jetty -keyalg RSA -sigalg
    SHA1withRSA -validity 1000 -keystore ui-ssl.keys
```

When keytool prompts for your name, enter your machine's fully qualified domain name (for example, host.domain.com). You also must enter a password for the keystore file and a password for the private key.

You can use any values you want for the -alias, -validity and -keystore parameters. We recommend always using -keyalg RSA. Instead of -sigalg SHA1withRSA, you can use -sigalg MD5withRSA. See the keytool documentation for more information about generating key pairs.

Copy the keystore file to [install directory]\common\conf\keys. This is where Cyclone Activator stores its other keystore files.

Edit startup.xml file

You must add an HTTPS service in the startup.xml file in [install directory]\[build number]\conf.

Open the file for editing. Locate the following commented-out control node service element:

```
<Service
class="com.cyclonecommerce.clustercontroller.httpserver.HttpsServerStarter"
    param="port=443
           keystore=../../common/conf/keys/ui-ssl.keys
           keystoreType=JKS
           keystorePassword=keystorePassword
```

```
            keyPassword=keyPassword
            clientAuthentication=false"/>
```

Uncomment this section of the file.

This control node service element has a class attribute of com.cyclonecommerce.clustercontroller.httpserver.HttpsServerStarter. This element tells the server to run the UI over an HTTPS server using:

- Port 443
- Keystore file ../../common/conf/keys/ui-ssl.keys
- Keystore type of JKS
- Keystore password of keystorePassword
- Private key password of keyPassword
- No client authentication (see Client authentication on page 57)

Note that all parameters to HttpsServerStarter are specified in the single param attribute.The value of the param attribute is one or more name-value pairs. If a parameter value contains white space or other special characters, enclose the value in double quotes or single quotes. For example:

```
param="port=443
  keystore=../../common/conf/keys/ui-ssl.key
  keystoreType=JKS
  keystorePassword=&quot;a password with spaces&quot;
  keyPassword='another password with spaces'
  clientAuthentication=false"
```

Note the use of **&quot;** to specify double quotes.

The port, keystore, keystorePassword and keyPassword parameters are required. The other parameters are optional and are shown with their default values.

Client authentication

If the clientAuthentication parameter is set to **true** for the HTTPS service in the startup.xml file, the server requires the user's browser to send a certificate back to the HTTPS server. How a browser picks which certificate to send varies by browser. But the certificate must be a personal certificate with a private key owned by the browser user.

The HTTPS server must trust the certificate returned by the browser client. If a browser user has a CA-issued certificate for authentication, you only must trust the root CA certificates. If a browser user has a self-signed certificate, the user must export the certificate and public key to a file and give you the file. You then must import the certificate to your keystore, as the following example shows:

```
    keytool -import -keystore ui-ssl.keys -alias client1 -
    file client_cert.cer
```

Multiple private keys

If your keystore contains multiple private key entries, they must all have the same key password. The HTTPS server selects one at random to use as the server's SSL certificate. This is not a recommended practice, as it will not be clear to you or your clients what certificate will be used by the SSL server. If you want to replace an old private key with a new one, it is recommended that you use:

```
    keytool -delete -keystore ui-ssl.keys -alias keyAlias
```

❖ ❖ ❖

5 Navigating the user interface

The web-based user interface has been designed for ease of use. It employs many of the conventions found on popular web sites, and you navigate it just as you would any web site. Self-guiding wizards are used liberally to walk you through configuration tasks.

Concepts

- Online GUI introduction
- The toolbar on page 59
- Task and alert display latency on page 61
- Navigation aids for the trading engine on page 61
- Help for users of earlier versions on page 62
- Fields that disappear on some pages on page 68

Online GUI introduction

An animated introduction to the user interface is available online. To launch the tour, log on to the user interface and click the GUI introduction link on the Help menu. The tour introduces the major areas of the user interface and how to navigate them.

The toolbar

The primary navigation aid is the top toolbar.

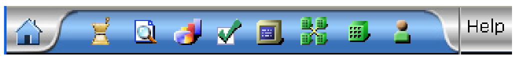


Figure 5. Toolbar example

The following table describes all toolbar elements. Elements, except Home, expand when you place the cursor over them to reveal task menus.

Icon	Name	Link description
	Home	Application home page for a dashboard view of system activity.

Icon	Name	Link description
	CSOS	Configure and view orders for compliance with the Controlled Substance Ordering System. This displays only when users have a CSOS license.
	Message tracker	Search and display trading activity records and documents.
	Reports	Configure user-defined reports.
		View the alert activity report .
✓	Alerts	View alerts.
	Alerts	Blinking icon indicates alerts requiring user attention. When there are no alerts, the icon changes to the alerts icon.
	System management	Manage miscellaneous system configurations.
	Peer network	Lets users configure a network for managing multiple trading engine clusters. This displays only when users have a peer network license.
	Trading configuration	Configure trading community entity profiles and other trading settings.
	Partners	Configure trading partner profiles.
	Users and roles	Configure users and permissions.
Help	Help	Product information and user documentation.

Task and alert display latency

Because of the way the server polls for status, tasks or alerts may continue to display on the tasks and alerts toolbar menu for up to several minutes after you have clicked on them and resolved the items. This also means tasks or alerts may not display immediately upon being triggered, but anywhere from some seconds to several minutes later. This is the result of a tuning adjustment that favors overall user interface performance and faster page loading. The trade-off for the improved performance is the task and alert display latency. For instance, if you satisfy an alert condition, the alert may continue to display on the toolbar for up to up to several minutes. The duration of the latency depends on the occurrence of the next interval when the server polls for task and alert status. At the next polling interval, the server finds the alert has been resolved and stops displaying it.

The display latency only occurs in the UI. It has no bearing on real-time writing of events to log files or sending of events via e-mail, if configured in the alerts.xml file in the system conf directory.

Navigation aids for the trading engine

The trading engine has graphics that help you manage trading profiles and configurations for communities and partners. When you place the cursor over an element, a red box illuminates the area. You can click any element to go to the named area.

Figure 6 shows the community navigation graphic, and Figure 7 shows the partner graphic. The graphics appear on the community or partner summary page and related pages.

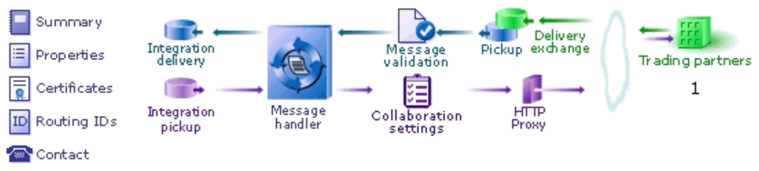


Figure 6. Community navigation graphic

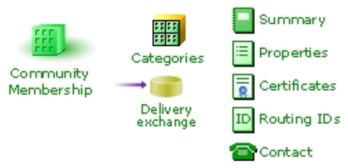


Figure 7. Partner navigation graphic

For more information see:

- Anatomy of a community profile on page 143
- Anatomy of a partner profile on page 148

Help for users of earlier versions

If you have used earlier versions of Cyclone software, you will notice significant differences between your legacy system and this application. Taking a few minutes to explore the user interface, you will find many of the same features and functions, but with a fresh look and feel.

Experienced users of Cyclone Activator will notice a change in the way the new application manages trading profiles. Where Cyclone Activator used **companies** as the entities representing your side of trading relationships, this application uses **communities**. This metaphor also envelopes all your trading partners. A community not only is the entity representing your organization or company, but your partners as well. Figure 8 illustrates this concept, which is reflected in the user interface. For more information about communities, see Trading configuration on page 141.

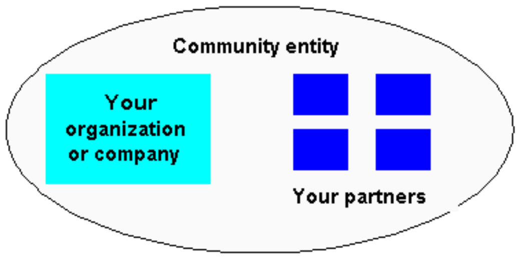


Figure 8. The trading engine view of a community entity

The following topics are to help users of earlier versions of Cyclone Activator locate functionality in the user interface of this updated application. The company and partner profile windows of version 4.2.x are related to areas of the new user interface.

Relating 4.2.x company profile to new system

The following relates the functionality in a 4.2.x company profile to its location in the user interface of 5.0 and later.

4.2.x Identity tab

Location of functionality in 5.0 and later:

To enter contact information for a new community, select **Trading configuration** > **Manage trading configuration** and click **Add a community**.

To change contact information, open the community summary page. Do this by selecting **Trading configuration** and the name of the community under **Recent communities**, or select **Trading configuration > Manage trading configuration** and click the community name. Once the community summary page is displayed, click **Contact** in the navigation graphic at the top of the page.

4.2.x Preferences tab

Location of functionality in 5.0 and later:

This version does not support sending alert or notification messages by e-mail.

Message backups are part of the configuration for delivery exchanges, which are what transports are called. To view or change a backup preference, open a community or partner summary page and do the following.

On a community summary page, click **Integration pickup**, **Integration delivery** or **Delivery exchange** in the navigation graphic at the top of the page. Click a delivery exchange and select the **Advanced** tab.

On a partner summary page, click **Delivery exchange** in the navigation graphic at the top of the page, click a delivery exchange and select the **Advanced** tab.

4.2.x Inbound Protocols tab

Location of functionality in 5.0 and later:

Delivery exchanges for receiving messages from partners are set up in community profiles.

On a community summary page, click **Delivery exchange** in the navigation graphic at the top of the page.

To view or change a delivery exchange for receiving messages, click a delivery exchange. To add a delivery exchange, click **Add a delivery exchange** to open the delivery exchange wizard. See The delivery exchange wizard on page 189.

For information about delivery exchanges, see Delivery exchanges on page 181.

4.2.x XML tab

Location of functionality in 5.0 and later:

Setting XPaths for senders and receivers in XML documents is part of the configuration of an integration pickup delivery exchange and a community delivery exchange for receiving messages.

To set up XPaths in a new integration pickup delivery exchange, open the delivery exchange wizard. See The delivery exchange wizard on page 189.

To view or change XPaths in an integration pickup delivery exchange, click **Integration pickup** in the navigation graphic at the top of a community summary page. Click the name of a delivery exchange and select the **From address** or **To address** tab.

To view or change XPaths in a community delivery exchange for receiving messages, click Delivery exchange in the navigation graphic at the top of a community summary page. Click the name of a delivery exchange and select the **From address** or **To address** tab.

For more information, see From address and To address tabs on page 291

4.2.x System Directories tab

Location of functionality in 5.0 and later:

A community retrieves messages for sending to partners using an integration pickup delivery exchange. On a community summary page, click **Integration pickup** in the navigation graphic at the top of the page.

A community routes messages received from partners to an integration delivery exchange. On a community summary page, click **Integration delivery** in the navigation graphic at the top of the page. If there is more than one exchange, the community uses the one at the top of the list.

You also can use post-processing scripts to direct inbound EDI, XML or binary messages to separate directories. See Manage file system integration on page 227.

4.2.x Integration tab

Location of functionality in 5.0 and later:

The following delivery exchanges are supported for routing messages received from partners to back-end systems: FTP, file system, JMS, MQSeries, ebXML message meta-data and web services API client. Post-processing also is supported.

To add or change an integration delivery exchange, click **Integration delivery** in the navigation graphic at the top of a community summary page. Click a delivery exchange to view or change or click **Add an integration delivery exchange**.

See Post-processing configuration details on page 222 for information about this option.

4.2.x Tuning tab

Location of functionality in 5.0 and later:

Tuning is part of the configuration of delivery exchanges.

On a community summary page, click **Integration delivery**, **Integration pickup** or **Delivery exchange** in the navigation graphic at the top of the page. Click a delivery exchange and select the **Advanced** tab.

On a partner summary page, click **Delivery exchange** in the navigation graphic at the top of the page. Click a delivery exchange and select the **Advanced** tab.

Relating 4.2.x partner profile to new system

The following relates the functionality in a 4.2.x partner profile to its location in the user interface of 5.0 and later.

4.2.x Identity tab

Location of functionality in 5.0 and later:

To enter contact information for a new partner, select **Partners** > **Add a partner**.

To change contact information, open the partner summary page. Do this by selecting **Partner** and the name of the partner under Recent partners, or select **Partners > Manage partners** and click the partner name. Once the partner summary page is displayed, click **Contact** in the navigation graphic at the top of the page.

The message handler area of the user interface manages message rerouting. On a community summary page, click **Message handler** in the navigation graphic at the top of the page. For information see Message handling on page 405.

4.2.x Preferences tab

Location of functionality in 5.0 and later:

The controls on the Preferences tab are located in various places in the new user interface.

Settings to preserve original file names or generate unique file names are part of delivery exchange configuration for partner delivery exchanges and community integration delivery exchanges. On a partner summary page, click **Delivery exchange** in the navigation graphic at the top of the page. On a community summary page, click **Integration delivery** in the navigation graphic at the top of the page.

The setting to compress documents a community sends is part of collaboration configuration. On a community summary page, click **Collaboration settings** in the navigation graphic at the top of the page. Click **Default settings** at the top left of the page.

The setting to allow or reject duplicate EDI messages received from partners is part of message validation configuration. On a community summary page, click **Message validation** in the navigation graphic at the top of the page.

Settings to resend messages when a community does not receive an expected receipt are part of collaboration configuration. On a community summary page, click **Collaboration settings** in the navigation graphic at the top of the page. Click **Default settings** at the top left of the page and select the **Reliable messaging** tab.

Settings to try again to send a message when a transport fails are part of the configuration of a delivery exchange for sending messages to partners and a community integration delivery exchange. On a partner summary page, click **Delivery exchange** in the navigation graphic at the top of the page, click an exchange and select the **Advanced** tab. On a community summary page, click **Integration delivery**, select an exchange and select the **Advanced** tab.

4.2.x Outbound Protocols tab

Location of functionality in 5.0 and later:

Delivery exchanges for sending messages to partners are set up in partner profiles.

On a partner summary page, click **Delivery exchange** in the navigation graphic at the top of the page.

To view or change a delivery exchange for receiving messages, click a delivery exchange. To add a delivery exchange, click **Add a delivery exchange** to open the delivery exchange wizard. See The delivery exchange wizard on page 189.

For information about delivery exchanges, see Delivery exchanges on page 181.

4.2.x Firewall tab

Location of functionality in 5.0 and later:

A community can route all outbound messages through an HTTP proxy server. On a community summary page, click **HTTP proxy**. See Outbound HTTP proxy on page 305 for information.

A community also can connect to a partner through an HTTP proxy if the partner requires this. On a partner summary page, click **Delivery exchange**, click an HTTP or HTTPS delivery exchange and select the **Proxy** tab.

4.2.x Security tab

Location of functionality in 5.0 and later:

Settings for signing and encrypting messages are part of collaboration configuration.

On a community summary page, click **Collaboration settings** in the navigation graphic at the top of the page. Click **Default settings** at the top left of the page. For information see Collaboration settings on page 367.

4.2.x Binary Directories tab

Location of functionality in 5.0 and later:

A community can route binary messages received from partners to a specific directory using a post-processing script. For information see Manage file system integration on page 227.

Fields that disappear on some pages

On some pages in the user interface, you might notice that some fields have disappeared. This occurs when you place the cursor over any part of the top toolbar. The disappearing fields are noticeable especially on the Message tracker page. The fields disappear only on some pages and only when the cursor is over the toolbar. The fields reappear when the cursor is moved away from the toolbar. This is an intentionally designed behavior to ensure that toolbar menus display properly when expanded menus overlap page fields. The display issue is solely browser-related and has no effect on the operation of Cyclone Activator.

❖ ❖ ❖

6 UI usage with proxy servers

This topic is for system administrators who must deploy Cyclone Activator, BEA WebLogic Edition in a network environment where users' browsers make HTTP connections through proxy servers.

Concepts

Deployment in proxy environment

Cyclone Activator has a web-based user interface served by a built-in servlet container. When the server is running it listens for HTTP connections on port 6080.

Cyclone Activator can be deployed in a network environment where proxy servers are used to enhance security, caching or logging. Two typical proxy server implementations are described: forward proxy and reverse proxy.

Forward proxy

In a forward proxy configuration, the web proxy server is used within a company's local area network behind a firewall or in the DMZ. Path A in Figure 9 illustrates that browser users connect to internal and external servers via a proxy server behind a firewall. Usually as a matter of policy, all browsers in the company are configured to go through the proxy server to connect to internal and external web servers. A browser can be configured to bypass the proxy server (path B in Figure 9), but this probably would go against policy.

The web proxy server might be set up inside the firewall, as in Figure 9, or in the DMZ. If inside the firewall, the proxy server is configured to route internal HTTP traffic directly to the servers. It does this based on the domain name or IP subnet. If in the DMZ, the browser is configured to route HTTP to the proxy when an Internet server is detected.

Cyclone Activator can be deployed in a forward proxy environment. While this does not require modifying browsers, adjustments are required for the proxy server.

The proxy server needs to be configured to restrict hosts to Cyclone Activator domains. It also must be configured to provide direct access to internal web servers.

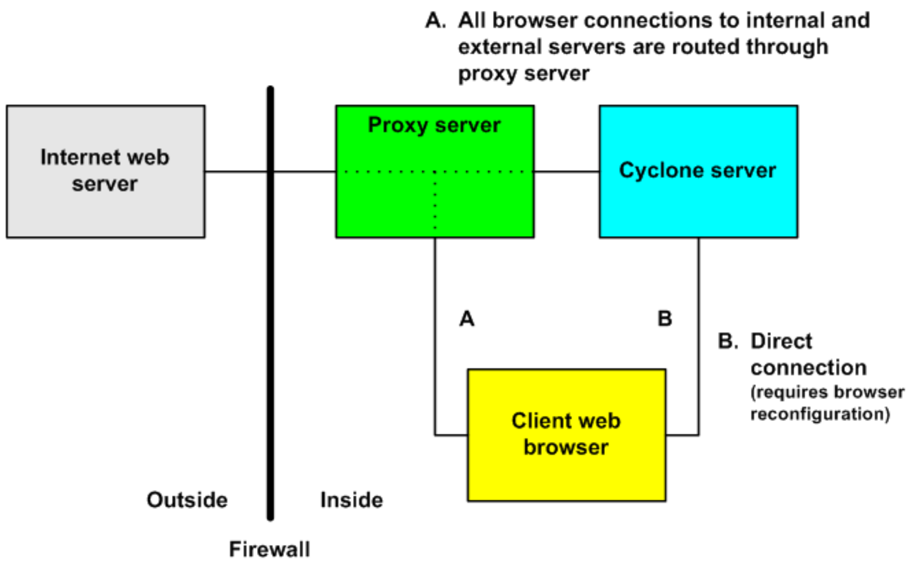

A. All browser connections to internal and external servers are routed through proxy server
Internet web server
Proxy server
Cyclone server
A
B
B. Direct connection (requires browser reconfiguration)
Client web browser
Outside
Inside
Firewall

Figure 9. Forward proxy configuration

Reverse proxy

In a reverse proxy configuration, the web proxy server is in the DMZ (Figure 10). It provides a secure path for external client browsers through the firewall to the internal web server. The external users address their browsers to the proxy host name and port number and might use the secure HTTP protocol, HTTPS. The proxy server translates the external browsers' requests to the host name and port number on the inside of the firewall.

To deploy Cyclone Activator in a reverse proxy environment, configure the proxy with reverse mapping. Consult with the proxy server administrator or see the proxy documentation for how to configure the mapping. For example, consider a situation where the server runs on HostA port 6080 and the proxy server runs on HostB port 8080. In this case external browsers would use the following URL to connect to the server on the inside: http://HostB.collaborationsoftware.com:8080.

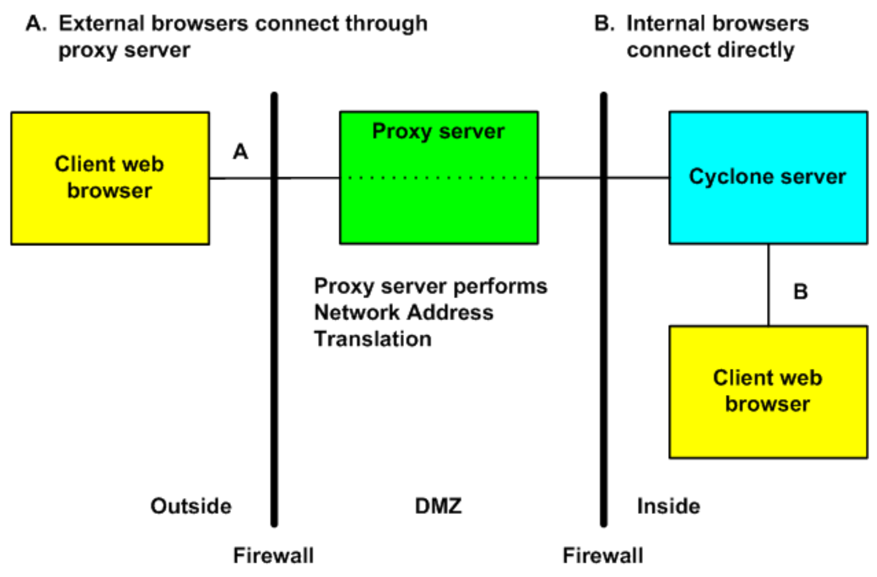

A. External browsers connect through proxy server
B. Internal browsers connect directly
Client web browser
A
Proxy server
Cyclone server
Proxy server performs Network Address Translation
B
Client web browser
Outside
DMZ
Inside
Firewall
Firewall

Figure 10. Reverse proxy configuration

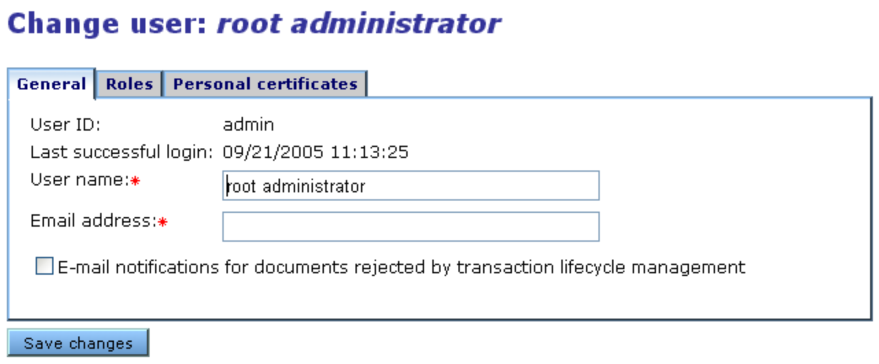
❖ ❖ ❖

7 User administration

Users and roles is the area of the user interface for adding and managing users. Roles define the permissions users have for performing tasks. Roles can be defined with few or many permissions. Each user should be assigned at least one role, although it is possible to assign multiple roles to a single user.

You also use this area to change users' passwords and manage global settings, such as session time-out intervals.

For users with CSOS capability, a personal certificates tab is provided for associating a user with certificates. See CSOS orders on page 503.


Change user: *root administrator*

User ID:	admin
Last successful login:	09/21/2005 11:13:25
User name:*	root administrator
Email address:*	

☐ E-mail notifications for documents rejected by transaction lifecycle management

Figure 11. Change user page in users and roles area of the user interface

Concepts

- The admin user on page 74
- Global user settings on page 76

Procedures

- Add, change, delete users on page 74
- Change password on page 75
- Add, change, delete roles on page 75
- Unlocking a blocked user on page 78

The admin user

The default system user has a user ID of **admin**. Its initial user name and password are **root administrator** and **Secret1**. This user is assigned to a role named **admin**, which has permissions to perform all functions. You cannot view or change the permissions of the admin role. You cannot delete the admin role or the admin user.

After logging on the first time with the admin user, we recommend immediately changing the password.

Although you can change the user name and password of admin and use it as a system administrator user, we recommend creating another user for that purpose and reserving the admin user as a backup.

Add, change, delete users

The users and roles area of the user interface has links for adding, changing and deleting users.

Note: If your software license allows users to have certificates, see CSOS orders on page 503.

To add a user	Select **Users and roles** > **Add a user**. Complete the fields, choose a role for the user and click **Add this user**. If the role you want is not available, you can create a role and add the user to it later.
To change a user	Select **Users and roles** > **Manage users**, select a user, make the changes you want and click **Save changes**.
To delete a user	Select **Users and roles** > **Manage users**, select a user and click **Delete this user**. Rather than deleting, you can disable a user by clearing the **Enable this user** check box.

The following are some tips for managing users.

- User names and passwords are case sensitive.
- Selecting the request e-mail notification check box makes the user eligible to receive alerts and reports by e-mail.

- Clearing the enable this user check box deactivates a user so the user no longer can log on. You can use this option when you want to suspend, but not delete, a user.
- When adding a user, remember to assign a role. A user without a role can log on, but can do nothing else.
- One user can log on to multiple browser sessions at the same time using the same user ID and password.
- The total number of browser sessions that can run concurrently is controlled by the user license. You can check the maxUserSessions element in the license.xml file for the authorized number. The file is in [install directory]\[build number]\conf. Although you can view the license file, do not try to change it, as that renders the application inoperable.

Change password

Any user with permissions for managing users can change his own password and the passwords of others.

To change a password, select **Users and roles > Manage users** and click a user name. On the user's page, click **Change this user's password**. Type the new password twice in the fields and click **Save changes**. The new password is effective the next time the user logs on.

Add, change, delete roles

The users and roles area of the user interface has links for adding, changing and deleting roles.

Roles are the permissions that define what users are allowed to do in the user interface. Roles are named and assigned to users. An administrator role typically has permissions to perform all tasks. The system role named admin has all permissions. You can create roles that have many or few permissions.

To add a role	Select **Users and roles > Add a role**. Type a name for the role and, optionally, a description. Review the list of permissions and select the ones you want for the role. Click **Add this role** when done.

To change a role	Select **Users and roles** > **Manage roles**. Click the name of the role to change, make the changes you want and click **Save changes**.
To delete a role	Select **Users and roles** > **Manage roles**. Click the name of the role to delete, scroll to the bottom of the change role page and click **Delete this role**.

Global user settings

Change global user settings on the users and roles menu opens a page that lets you configure user interface session settings affecting all users.

Only users assigned to a role with the "manage users and roles" permission can view or change global user settings.

This page has two tabs: Session management and User security. The following topics describe the fields on each tab.

Session management tab

Maximum session length (minutes)

The number of minutes a session can be idle before the system logs off the user.

Login retries

The number of times a user can try unsuccessfully to log on before the system locks out the user. This is a safeguard against possible efforts by unauthorized users to access the system.

Lockout length (minutes)

The interval in minutes that a lockout is in effect. When the lockout expires, the user can try again to log on. If you want to unlock a user immediately see Unlocking a blocked user on page 78.

Days to save audit events

The number of days to retain records of user activity in the database. This data is not accessible in the user interface or a log file.

Allow a user to have concurrent browser sessions

Selecting this allows all users to log on multiple times to the user interface simultaneously. When unchecked, each user can have only a single browser session.

If you select this, make sure the maxUserSessions element in the license.xml file in the system conf directory can support many concurrent user sessions.

User security tab

If Cyclone Activator is part of an external single sign-on (SSO) system implementation, the user security tab does not display. The external SSO system controls these settings.

Minimum user ID length

The minimum number of characters allowed for user IDs. The length can range from a minimum of 5 characters to a maximum of 20 characters.

A user ID can be any combination of alphanumeric characters and is case sensitive.

If you change the minimum user ID length, the new minimum is enforced only for new users. IDs of users who pre-date the change remain valid.

Minimum password length

The minimum number of characters allowed for user passwords. The length can range from a minimum of 6 characters to a maximum of 20 characters.

A password can be any combination of characters and is case sensitive. At least one character must be a number. In addition, at least one non-numeric character must be upper case and at least one non-numeric character must be lower case.

If you change the minimum password length, the new minimum is enforced for users added after the change. Passwords of users who pre-date the change remain valid. However, the new minimum is enforced when a password is changed.

Minimum change count before password can be reused

The number of times a user must change a password before a previous password can be re-used.

Elapsed days before password can be reused

The number of days that must pass before a user can re-use a password.

Days password remains valid before it must be reset

The number of days a password is valid before it must be changed.

Elapsed days before disabling an inactive user

The number of days before an inactive user's account is disabled. A disabled user can be re-activated.

Force new users to reset their passwords upon initial logon

Selecting this compels all new users to change their passwords after logging on the first time.

Unlocking a blocked user

You can unlock a user who has been blocked from logging on to the user interface. A lockout occurs after a user repeatedly fails to log on with an incorrect user name or password. You can immediately unlock a user without waiting for the lockout interval to elapse.

The utility to use is unlockuser.cmd in [install directory]\[build number]\bin. Run the utility without parameters in a UNIX console or Windows command window.

The utility prompts for the password of the admin user and the user ID of the user to unlock.

❖ ❖ ❖

8 Activity tracking and logs

There are a number of ways to monitor system activity. Methods are available through the user interface and log files. The user interface methods are easier to use and understand than the log files, which are designed for software developers or advanced users.

Note: If you want to send messages about trading engine events by e-mail, see The alerts.xml file on page 106.

Concepts

Procedure

User interface tracking

The user interface has tools for monitoring various types of system activity.

Home page

The home page is designed as a dashboard of system-wide activity. It warns of alerts requiring user attention, reports summary information about trading activities, reports system status and lets you perform quick searches of messages.

System monitor

The system monitor reports real-time activity for the trading engine. You can launch the system monitor after starting Cyclone Activator server and leave it running.

The system monitor can be invoked from various places, including the home page, and appears in its own browser window. Aside from the home page, you can launch the system monitor from the system management menu or page.

System management

System management is an area for managing system-related configurations.

At the bottom of the system management page is a link for “generate cluster thread dumps.” This generates a file of system statistics that can be useful in troubleshooting. Technical support may ask you to generate and send one of these files while investigating a system problem for you. These files do not contain user consumable information.

Statistics monitor

The statistics monitor lets you build charts for tracking system performance indicators in real time. For instance, you can set up a chart for tracking heap memory usage, database response time, message consumption and production rates.

There are many types of charts you can construct. The charts provided by default are examples of the types you can set up. You can organize charts on tabs, one or many charts per tab.

To explore, select **Systems management > Statistics monitor**. The statistics monitor uses a Java applet. The first time you use the monitor, you are prompted to accept the applet. This occurs only the first time.

Here are some tips for navigating and using the statistics monitor:

- Click a chart to select it and right-click to open a menu for changing properties, printing or saving the chart to a PNG file.
- Immediately below a chart is displayed the values for the last data point. You can view the values for other data points by clicking the points.
- To build a chart, select **Layout > New chart**. Note the fields for naming the chart, selecting the time scale for the horizontal axis in minutes, hours and days, and setting values for the vertical axis. You can choose to show or hide data points along the graph line.
- When setting up a new chart, there are many statistics categories in the **Available** list. You can expand each category to display more options. Some items display as green; others as black. The green

items represent data that are persisted to the database; black items are not persisted. The difference means a new chart already may have a history of data to display (green) versus a new chart where data starts building after inception of the chart (black). You can turn persistence on and off for the data elements in the monitoringconfig.xml file in [install directory]\[build number]\conf.

Message tracker

Message tracker is the primary tool for tracking the messages traded between you and your trading partners.

Message tracker lets you search for messages by various conditions, including sending and receiving parties, processing status and dates. Once the system finds messages matching your search conditions, you can view trading history details and document payloads. You also can reprocess documents.

Message tracker also lets you save searches so you can perform the same searches repeatedly without having to set up the conditions again.

For more information see Message tracker on page 411.

Alert activity report

The alert activity report can display categories of information regarding processing error, fatal, rejected and notification events. Select **Reports > Alert activity report**.

External monitoring of server status

A servlet that runs on the trading engine can return a default server status message or custom page when a client performs an HTTP GET. External content switches and system monitors can hit the servlet periodically. The response indicates the trading engine server and all network components in between are operational.

To use this feature, the server must be running and a community profile must be set up. Also, for Cyclone Interchange at least one node must be started. The community profile must have a delivery exchange for receiving messages from partners that uses an embedded HTTP server. For information about configuring a profile, see The community profile on page 142.

You can retrieve the default server status message with a URL in the following format:

http://host:port/ServerStatus

Host is the name or IP address of the computer running the trading engine server. Unless it has been changed, use **4080** as the **port**. This is the default port of embedded HTTP servers (see Embedded transport servers on page 167).

You can point a browser to the URL to manually inspect the status message. The default message is **ServerStatus=OK**. To refresh the page, press **Control** while selecting **Refresh** to force the browser cache to clear.

Instead of the default message, you can design a status file with any content you like. When you use the server status URL, your file is displayed. You can use this feature to have external systems generate a more comprehensive or sophisticated status page.

The name and path of the custom file are controlled by the filereg.xml file in [install directory]\[build number]\conf. The following are the default settings:

```
<File name="serverstatus.html"
path="conf/serverstatus.html" />
```

To use the default configuration for the custom file, create an HTML document and save it as **serverstatus.html**. Copy the file to [install directory]\[build number]\conf.

You do not have to use an HTML file. You can use any file type you want and you can use any file name. If you do, change the **path** attribute in filereg.xml to conform to your file name and location. However, do not edit the **name** attribute. This value must remain as **serverstatus.html**.

Log file tracking

The system writes many kinds of log files to its logs and other directories. For the most part, these logs are troubleshooting tools for software developers and not intended for end users. Experienced users, however, might gain insights into processing activity by examining certain of these files. Log files contain a complex array of data that takes practice to interpret.

Detailed information about system events that users might find useful and how to manage and route them to various log files is in Events system on page 91.

The following topics describe the log files.

Event log

This log contains selected events from the event subsystem. These are events related to message processing activity. If you want to use a log to monitor processing activity, this is the log you may want to examine. The log is **server_events.log** and is written to [install directory]\[build number]\logs.

System logs

System logs contain formatted, time-stamped information reported by various components of the application. The logs, intended for use by software developers in troubleshooting, are not supported for end users. Cyclone Activator uses the Apache Jakarta Project's log4j framework to format and manage the logs, which are generated by each Java virtual machine during runtime. The log4j.properties file in [install directory]\[build number]\conf can be edited to generate debug level events in log files.

For information about Apache logging services, see http://logging.apache.org/log4j/docs/index.html.

See Troubleshooting with log4j file on page 87 for information about changing and using the file.

The system names the logs based on the names of the source JVM node. They are written to [install directory]\[build number]\logs. The logs are:

- **hostname.ex.log** is created by the system Executive node.
- **server.log** reports processing activity of the trading engine.

System logs can report four levels of events. These are, in order of verbosity:

- **Error** messages indicate a possibly serious error affecting service.
- **Warn** messages typically have operational significance, but might not affect service.
- **Info** messages provide general processing information that could be useful in troubleshooting.
- **Debug** messages usually have much detailed information that can be useful in troubleshooting. This level should be turned on only when necessary, as it can degrade system performance and use large amounts of disk space.

The four logging levels are cumulative. Error messages are included at the warn level, both of which are included at the info level, and everything is logged at the debug level. The debug level also produces further details about processing.

System statistics logs

All logs file appended with **stats.log** are Java Management Extensions (JMX) monitoring and statistics logs. There can be many of these files, depending on how many processing nodes are active.

Although not for end users, these logs are an aid in troubleshooting. If you contact technical support for help, a technician may ask you to send copies of these files.

User Interface logs

These logs are created by the user interface. Like system logs, they do not contain information useful to the typical end user. These logs are written to [install directory]\[build number]\logs\ui. The logs are:

- access.log
- core.log
- error.log
- sitemap.log
- ui.log

The log file names use as a prefix the name of the computer on which Cyclone Activator software is installed. The names have the following format: **hostname_cn_access.log.000001**. The **cn** stands for control node. The trailing number (000001) identifies rolling logs. Once a log reaches a certain size, events write to another log (000002). This rolling occurs up to five times before events again start writing to the first log file.

HTTP server logs

These logs are written by the embedded HTTP server when a control node or service node is started with the -Ddebug startup option.

Other logs

These include Cyclone_InstallLog.log, which contains information about the initial system installation, and dbConfigurator.log, which contains messages logged by the database configuration utility. Both of these logs write to [install directory]\common\logs.

Real-time viewing of log files

Use this procedure to use a utility that lets you view activity as the system writes it to any log file. This procedure explains how to use it in a Windows environment. Usage on UNIX is similar.

Note: If your operating system is Windows Server 2003 SP1, you need to download the Windows Server 2003 Resource Kit Tools and use the tail.exe from that package rather than the tail.exe in [install directory]\[build number]\bin. The download link is http://www.microsoft.com/downloads/details.aspx?FamilyID=9d467a69-57ff-4ae7-96ee-b18c4790cffd&displaylang=en.

Steps

1 Choose the log file you want to monitor. Logs are in the logs directory of the installation hierarchy.

For example purposes, we choose to monitor a node events log.

2 Open a command window. From [install directory]\[build number]\bin, run the tail command in the following format:

tail -f path logname

In our example, the command is:

tail -f c:\[install directory]\[build number]\\logs\hostname_te_events.log

Set up tail as a Windows option

Use this procedure to set up the tail.exe utility in a way that lets you view real-time logging activity in Windows when you right-click a log file name and select a tail option.

Steps

1. Create a directory on your local drive. For example, c:\tailoption.

   It is preferred to create a separate directory rather than a subdirectory in Cyclone Activator root directory.

2. Copy **tail.exe** from [install directory]\[build number]\bin to the directory you created in the previous step.

3. In Windows Explore, select **Tools** > **Folder Options**.

4. Click the **File Types** tab.

5. Scroll down to the **LOG** extension in the list of file types. Select it and click **Advanced**.

6. Click **New**.

7. In the Action field, type something descriptive like **Tail**.

8. In the Application used to perform action field, type the path of the tail.exe file you copied in step 2. For example, c:\tailoption\tail.exe.

9. Click **OK**.

10. In the Actions area, select **Tail** or whatever you typed in step 7 and click **Edit**.

11. In the Application used to perform action field, type **-f** between **tail.exe** and **"%1"** as follows:

    Before: C:\tailoption\tail.exe "%1"

    After: C:\tailoption\tail.exe -f "%1"

12. Click **OK**.

13. Optionally, click **Set Default** to make **Tail** or whatever you typed in step 7 the default action for log files.

14. Click **OK**.

15. Click **Close**.

16. When Cyclone Activator server is running, navigate to [install directory]\common\logs, right-click a log file, select **Tail** or whatever you typed in step 7, and then view the log file as it is being written.

Troubleshooting with log4j file

The properties in the log4j.properties file control the level of messages written to system log files. By changing levels, you control the volume of messages written to the logs. Changing levels can be useful in troubleshooting.

System logs contain formatted, time-stamped information reported by various components of the application. The logs, intended for use by software developers in troubleshooting, are not supported for end users. Cyclone Activator uses the Apache Jakarta Project's log4j framework to format and manage the logs, which are generated by each Java virtual machine during runtime. The log4j.properties file in [install directory]\[build number]\conf can be edited to generate debug level events in log files.

For information about Apache logging services, see http://logging.apache.org/log4j/docs/index.html.

You do not need to restart the server after changing levels for message categories unless you remove an entry rather than change its level.

CAUTION: Using log files to monitor processing or troubleshoot requires advanced knowledge of the system.

The message levels, from lowest to highest verbosity, are:

Error

Error messages indicate a possibly serious error affecting service.

Warn

Warn messages typically have operational significance, but may not affect service.

Info

Info messages provide general processing information that could be useful in troubleshooting.

Debug

Debug messages usually have much detailed information that can be useful in troubleshooting. This level should be turned on only when necessary, as it can degrade system performance and use large amounts of disk space.

The four logging levels are cumulative. Error messages are included at the warn level, both of which are included at the info level, and everything is logged at the debug level. The debug level also produces further details.

Changing the log level of a category changes the level of all categories beneath it that do not have a logging level explicitly specified. For example, setting category “com.foo” to debug also sets “com.foo.fum” to debug unless there is an explicit entry for “com.foo.fum.”

Commonly used categories, from general to more specific, are described in the following list. To troubleshoot some issues, technical support may ask you to add even more specific categories not listed here.

Once a category has been set to a different level (for example, from info to warn), the level remains in effect until the category is set to something else. Simply removing or commenting out the category does not reset its level, unless the server is restarted. For this reason, if you add a category and set it to a verbose level like debug, we recommend resetting it to the info level when you are done rather than removing the added category. Otherwise, you would have to wait until the server is restarted for the verbose logging to stop.

You can change the levels of the following message categories.

General message categories

log4j.category.com.cyclonecommerce=info

The main message category. Includes all system log messages.

log4j.category.com.cyclonecommerce.persistence=info

Messages related to persistence.

log4j.category.com.cyclonecommerce.util=info

Messages from utility classes.

log4j.category.com.cyclonecommerce.clustercontroller=info

Messages related to system startup and clustering.

log4j.category.com.cyclonecommerce.alerts.AlertDefinitionsManager=info

Messages related to Alert system activity

Trading engine messages

log4j.category.com.cyclonecommerce.tradingengine=info

Messages related to the handling and routing of messages within the trading engine core.

log4j.category.com.cyclonecommerce.crossworks=info

Messages related to certificate and cryptographic operations.

log4j.category.com.cyclonecommerce.ediintmsg=info

Messages related to EDIINT messages (AS1, AS2, AS3).

log4j.category.com.cyclonecommerce.messageprotocols=info

Messages related to protocol sender and receiver operations.

log4j.category.com.cyclonecommerce.webservices=info

Messages related to ebXML and SOAP-based messaging.

Transport-related trading engine messages

log4j.category.com.cyclonecommerce.tradingengine.transport.system=info

Messages related to the transport subsystem (consumers and producers).

log4j.category.com.cyclonecommerce.tradingengine.transport.Polling=info

Messages specifically related to transport polling. Set to debug to log a message each time a transport is polled (every 60 seconds by default).

log4j.category.com.cyclonecommerce.tradingengine.transport.http=info

Messages related to the HTTP client and the embedded HTTP server.

log4j.category.com.cyclonecommerce.tradingengine.transport.ftp.SimpleDebug=info

Messages related to high-level operation of the FTP client. This is useful in debug mode for finding common FTP problems. See the next category for more verbose FTP debugging.

log4j.category.com.cyclonecommerce.tradingengine.transport.ftp=info

Messages related to low-level operation of the FTP client. In debug mode this produces very verbose messages. See the previous category for simpler and more readable FTP debugging.

log4j.category.com.cyclonecommerce.tradingengine.transport.email.smtp=info

Messages related to SMTP delivery exchange.

log4j.category.com.cyclonecommerce.tradingengine.transport.email.pop=info

Messages related to SMTP/POP delivery exchange.

Database messages

These are messages related to the database interface, Kodo. Typically these properties should only be changed under direction of technical support. Kodo documentation is available at http://www.solarmetric.com/jdo/Documentation.

```
log4j.category.kodo.Tool=warn
log4j.category.kodo.Runtime=error
log4j.category.kodo.Remote=warn
log4j.category.kodo.DataCache=warn
log4j.category.kodo.MetaData=warn
log4j.category.kodo.Enhance=warn
log4j.category.kodo.Query=warn
log4j.category.kodo.jdbc.SQL=warn
log4j.category.kodo.jdbc.JDBC=warn
log4j.category.kodo.jdbc.Schema=warn
```

❖ ❖ ❖

9 Events system

Cyclone Activator generates, publishes and routes defined events related to system activity.

Events are published in various ways: in log files, the user interface and by e-mail. These are events as cataloged in Event tables on page 112.

The events.xml file on page 92 describes how events are published to log files and JMS routers.

The alerts.xml file on page 106 describes publishing events about the trading engine to the user interface and by e-mail.

The events.xml and alerts.xml file have default settings for publishing events without user intervention. Both files, however, are configurable to expand or reduce the volume of published events.

Concepts

Event levels

A “level” indicates the significance of an event. Levels rank events from low to high importance. There are four levels, which are listed in order of lowest to highest importance: Low, High, Warning and Error.

Event levels provide different degrees of information. Low level events contain the most detailed information and high level events contain the most general information. The warning and error levels do not correspond to a level of detail, but do infer degrees of importance.

The events.xml file

The events.xml file in [install directory]\[build number]\conf, manages publishing events to log files and JMS routers. You must understand this file if you want to customize event configurations.

Each event contains standard information, including:

- Event type
- Event level
- Time stamp
- Name and address of the node that spawned the event
- Name of the application that spawned the event

An event also might have specific information in the form of meta-data content or extended event content.

The events.xml file that manages logged events is [install directory]\[build number]\conf. You can edit this file to control what events are generated and where they are published. By default a certain set of events — by no means all — are configured in events.xml to publish to system log files in the logs directory (see Log file tracking on page 82). Using event routers you also can publish events to a JMS queue.

In addition to using event routers that write events to log files or pass events to some other process, event filters provide a way for determining what events are delivered to a router. Events can be filtered based on type or level. You can build complex filters from multiple simple filters.

CAUTION: Before editing the events.xml file, we recommend making a copy of it in case you want to restore the default configuration for events.

The default configuration of events.xml is to write an events log file for each system node to the logs directory. This results in the generation of at least one but possibly more log files that, for Cyclone Activator, are named in the following format: hostname_te_events.log. For Transaction Director, the log name has the following format: hostname_tx_events.log.

One of these event logs is the system control node events log. By default the name of the control node is the host name of the computer. The name of the events log file for the control node looks like this: hostname_cn_events.log.

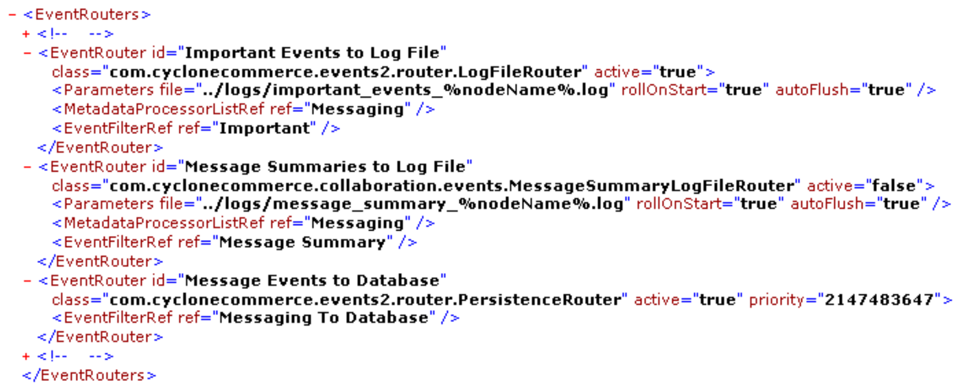
Be default events.xml is configured only to write certain events to events log files.

Figures in the following topics showing sections of the events.xml file are depicted as viewed in Internet Explorer. The display might look different on your system depending on the viewer or editor you use.

EventRouters section of events.xml

The top portion of the events.xml file is for event routers. Figure 12 is an example of the default **EventRouters** section of the file.

For details on how to configure routers, see Log file event router on page 98 and JMS event router on page 100.



```
- <EventRouters>
  + <!-- -->
  - <EventRouter id="Important Events to Log File"
      class="com.cyclonecommerce.events2.router.LogFileRouter" active="true">
      <Parameters file="../logs/important_events_%nodeName%.log" rollOnStart="true" autoFlush="true" />
      <MetadataProcessorListRef ref="Messaging" />
      <EventFilterRef ref="Important" />
    </EventRouter>
  - <EventRouter id="Message Summaries to Log File"
      class="com.cyclonecommerce.collaboration.events.MessageSummaryLogFileRouter" active="false">
      <Parameters file="../logs/message_summary_%nodeName%.log" rollOnStart="true" autoFlush="true" />
      <MetadataProcessorListRef ref="Messaging" />
      <EventFilterRef ref="Message Summary" />
    </EventRouter>
  - <EventRouter id="Message Events to Database"
      class="com.cyclonecommerce.events2.router.PersistenceRouter" active="true" priority="2147483647">
      <EventFilterRef ref="Messaging To Database" />
    </EventRouter>
  + <!-- -->
  </EventRouters>
```

Figure 12. EventRouters section of events.xml

EventRouter attributes

The **EventRouters** element can contain any number of **EventRouter** elements, each defining an event router to be installed in the event system. Each EventRouter element must have an **id** attribute and a **class** attribute. Other attributes are optional.

- **id** must specify a unique identifier for the router. If the configuration contains more than one EventRouter element with the same identifier, only the last router with the same identifier is installed.
- **class** must specify the full name of a class that implements the com.cyclonecommerce.events2.router.EventRouter interface.
- **active** is optional and specifies whether the event router should be made active when installed. Only an active router is started when the server starts. The value of the active attribute must be **true** (the default) or **false**.

- **priority** is optional and specifies the priority of the router relative to other routers. The value of the priority attribute must be a positive integer. The system event dispatcher sends events to routers in priority order of highest to lowest. The lower the value of the priority attribute, the higher the priority, with **1** the highest priority. If multiple routers have the same priority, events are dispatched to those routers in no particular order. If the priority attribute is not specified, the priority for the router defaults to the middle between the highest and lowest possible integer value.

Parameters element

An **EventRouter** element can have a **Parameters** element that defines parameters specific to the particular event router class. The Parameters element can contain attributes or sub-elements.

EventFilterRef element

An **EventRouter** element can contain an **EventFilterRef** element that identifies the event filter to be applied to the event router. Only events that pass through the filter are passed to the event router. The EventFilterRef element, if present, must contain a **ref** attribute with the same value as an **id** attribute in an **EventFilter** element. If a EventFilterRef element is not defined, all events are passed to the event router.

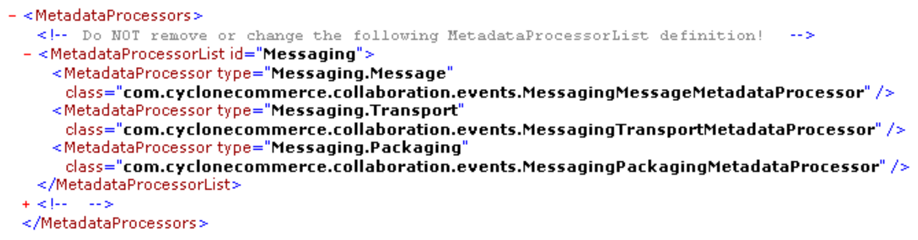
MetadataProcessorListRef element

An **EventRouter** element can have a **MetadataProcessorListRef** element that identifies what list of meta-data event content processors should be used to determine what meta-data processor to use for each event. The **ref** attribute must refer to a MetadataProcessorList element.

In the default configuration of events.xml, use MetadataProcessorListRef only when you want to publish events at the **Messaging** level. This element is not needed in other event router configurations.

MetadataProcessors section of events.xml

The **MetadataProcesssors** section follows the **EventRouter** section of the events.xml file. Figure 13 is an example of the default MetadataProcessors section.



```
- <MetadataProcessors>
    <!-- Do NOT remove or change the following MetadataProcessorList definition!  -->
  - <MetadataProcessorList id="Messaging">
      <MetadataProcessor type="Messaging.Message"
        class="com.cyclonecommerce.collaboration.events.MessagingMessageMetadataProcessor" />
      <MetadataProcessor type="Messaging.Transport"
        class="com.cyclonecommerce.collaboration.events.MessagingTransportMetadataProcessor" />
      <MetadataProcessor type="Messaging.Packaging"
        class="com.cyclonecommerce.collaboration.events.MessagingPackagingMetadataProcessor" />
    </MetadataProcessorList>
  + <!--  -->
  </MetadataProcessors>
```

Figure 13. MetadataProcessors section of events.xml

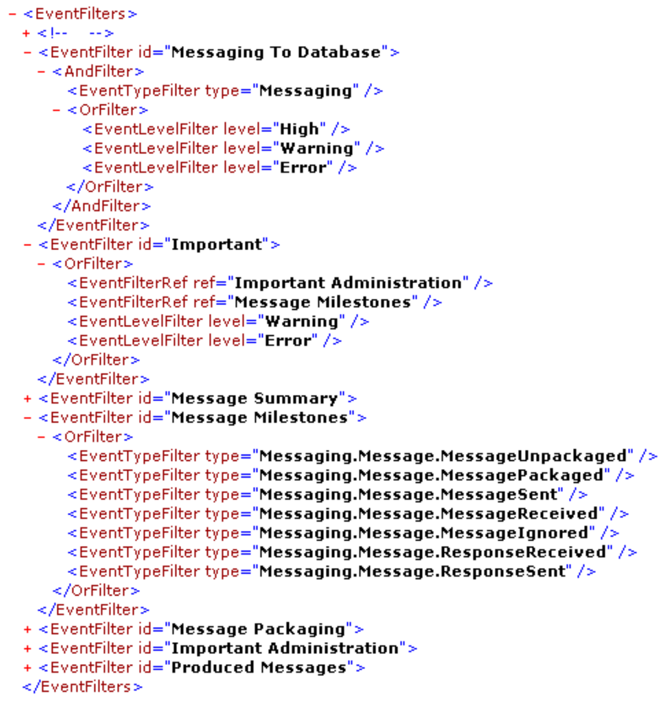
CAUTION: Although the following paragraphs explain this section of the file, it is imperative that you do not change it.

The **MetadataProcessors** element can have any number of **MetadataProcessorList** elements, each defining a list of meta-data processors. Each MetadataProcessorList element must have a unique **id** attribute and can contain multiple MetadataProcessor elements, but is not required to have any. Each MetadataProcessor element must have a **type** attribute specifying an event type and a **class** attribute specifying the name of a class that implements the com.cyclonecommerce.events2.metadata.MetadataProcessor interface.

When an event router needs a meta-data event content processor, it searches the referenced MetadataProcessorList element for the first MetadataProcessor element with an event type that matches the type of the event being processed by the event router. The event type in the MetadataProcessor element can be the exact type or a super-type of the router's event type. The MetadataProcessor elements are searched in the order specified.

EventFilters section of events.xml

The **EventFilters** section follows the **MetadataProcessors** section of the events.xml file. Figure 14 is a partial example of the default EventFilters section.



```
- <EventFilters>
  + <!--   -->
  - <EventFilter id="Messaging To Database">
    - <AndFilter>
        <EventTypeFilter type="Messaging" />
      - <OrFilter>
          <EventLevelFilter level="High" />
          <EventLevelFilter level="Warning" />
          <EventLevelFilter level="Error" />
        </OrFilter>
      </AndFilter>
    </EventFilter>
  - <EventFilter id="Important">
    - <OrFilter>
        <EventFilterRef ref="Important Administration" />
        <EventFilterRef ref="Message Milestones" />
        <EventLevelFilter level="Warning" />
        <EventLevelFilter level="Error" />
      </OrFilter>
    </EventFilter>
  + <EventFilter id="Message Summary">
  - <EventFilter id="Message Milestones">
    - <OrFilter>
        <EventTypeFilter type="Messaging.Message.MessageUnpackaged" />
        <EventTypeFilter type="Messaging.Message.MessagePackaged" />
        <EventTypeFilter type="Messaging.Message.MessageSent" />
        <EventTypeFilter type="Messaging.Message.MessageReceived" />
        <EventTypeFilter type="Messaging.Message.MessageIgnored" />
        <EventTypeFilter type="Messaging.Message.ResponseReceived" />
        <EventTypeFilter type="Messaging.Message.ResponseSent" />
      </OrFilter>
    </EventFilter>
  + <EventFilter id="Message Packaging">
  + <EventFilter id="Important Administration">
  + <EventFilter id="Produced Messages">
  </EventFilters>
```

Figure 14. EventFilters section of events.xml

The **EventFilters** element can have any number of **EventFilter** elements, each defining an event filter. Each EventFilter element must have an **id** attribute that specifies a unique identifier for the filter. Although more than one EventFilter element can have an **id** attribute with the same value, we recommend that each EventFilter element have its own **id**.

A filter defined by an EventFilter element only starts upon server startup when an EventRouter element references it directly or indirectly.

Each EventFilter element defines a single event filter. A single event filter can be simple or complex.

Simple filters

The following are simple event filters.

<AllFilter/>

A filter that passes all events.

<NoneFilter/>

A filter that passes no events.

<EventTypeFilter type="type"/>

A filter that passes all events of a specified type or any of its sub-types. The type attribute must be specified and its value must be the name of an event type defined in Event tables on page 112

For example, if you specify a type of **Messaging.Message**, all events with that prefix will pass through the filter.

<EventLevelFilter level="level"/>

A filter that passes all events at the specified level and above. The level attribute must be specified and its value must be, in order of least to most important, **Low**, **High**, **Warning** or **Error**.

For example, if you specify a level of **Low**, all Low level messages will pass through the filter, as well as all High, Warning and Error events. Conversely, if you specify a level of **Error**, only Error level events will pass.

<EventFilterRef ref="filterId"/>

A filter that refers to another event filter. Using this mechanism supports the re-use of event filters. The **ref** attribute must be specified and its value must equal that of an **id** attribute in an EventFilter element. Circular filter references are not allowed.

Complex filters

The following are complex event filters.

<NotFilter>

<!-- A single event filter definition must appear here. -->

</NotFilter>

A filter that passes all events that do not pass the simple or complex event filter defined within.

<AndFilter>

<!-- Any number of event filter definitions may appear here. -->

</AndFilter>

A filter that passes all events that pass all the event filters defined within.

<OrFilter>

<!-- Any number of event filter definitions may appear here. -->

</OrFilter>

A filter that passes all events that pass any of the event filters defined within.

Log file event router

The event system comes with a standard event router in events.xml that sends events to log files in the logs directory (Log file tracking on page 82). This outbound event router is implemented by the following Java class:

com.cyclonecommerce.events2.router.LogFileRouter

The default log file event router is configured in a system file for managing events. The file is [install directory]\[build number]\conf\events.xml. The following is an example of how the log file event router is set up in events.xml. For other details about this file, see The events.xml file on page 92.

```
<EventRouter id="Unique identifier of router"
class="com.cyclonecommerce.events2.router.LogFileRouter"
active="false">
<Parameters
   file="fileName"
   rollOnStart="boolean"
   autoFlush="boolean"
   gmt="boolean"
   maxFileSize="long"
   maxBackupFiles="integer"
   maxContentDepth="integer"
   rollTimes="hh:mm,hh:mm,hh:mm,…"
/>
<MetadataProcessorListRef ref="Messaging" />
<EventFilterRef ref="Important" />
</EventRouter>
```

The MetadataProcessorListRef element references an element in the MetadataProcessors section of events.xml. The EventFilterRef element references a filter in the EventFilters section of events.xml. The following explains the attributes of the EventRouter and Parameters elements in the router defintion.

EventRouter attributes

The following explains the attributes of the EventRouter element in a log file configuration in events.xml.

id

A unique identifier of the router.

class

Specifies the name of the Java class that implements the router interface.

active

Specifies whether the router should be made active when the server starts.

Parameters attributes

The following explains the attributes of the Parameters element in a log file configuration in events.xml.

file

Specifies the name of the event log file.

rollOnStart

Specifies whether the log file should be rolled when the log file router is started. Rolling a log file means starting a new log file, saving the current log with an extension of **.1** to the file name and renaming any older log files by incrementing the extension by 1. For example, a file with an extension of .1 is renamed **.2** , and so on. The default value is **false**. This attribute is optional.

autoFlush

Specifies whether events are written immediately to the log file. Using a value of **true** is handy when tailing the log and you do not want to wait before events are displayed. The default value is **false**, which causes a delay in writing events to the log, a lag that varies depending on the operating system. This attribute is optional.

gmt

Specifies whether time stamps should be GMT time. If not, time stamps use the local time zone. The default value is **false** if not specified. This attribute is optional.

maxFileSize

Specifies the maximum size of a log file before it is rolled. The size is specified in bytes. The letters K, M or G (case insensitive) can follow a numerical value to specify kilobytes, megabytes or gigabytes, respectively. This option only applies when logging to a file (not standard output). The default value is the maximum file size supported by the operating system. This attribute is optional.

maxBackupFiles

Specifies the maximum number of rolled log files to retain. This is in addition to the current log file. The default behavior is to retain all archived log files. This attribute is optional.

maxContentDepth

Specifies the maximum depth of extended content to display. A value of **0** displays no extended content, a value of **1** displays the top level of extended content, and so on. The default behavior is to display all extended content. This attribute is optional.

rollTimes

Specifies when to roll the log, regardless of file size. Times are specified in hours and minutes, using a 24-hour clock. If the value of the **gmt** attribute is true, the times in this attribute are assumed to be GMT times; otherwise, they are assumed to be local times. This option only applies when logging to a file (not standard output). The default behavior is not to roll at any specific times. This attribute is optional.

JMS event router

With the help of technical consultants, you can configure a router in events.xml to publish events to a JMS queue. To do so, you must have Java Message Service experience and a working JMS system.

Instructions for configuring a JMS event router are in the events.xml file.

Oracle AQ event router

With the help of technical consultants, you can configure a router in events.xml to publish events to an Oracle Advanced Queuing facility (Oracle AQ) queue. To do so, you must have Java Message Service experience and a working Oracle AQ system.

Instructions for configuring an Oracle AQ event router are in the events.xml file.

XML for JMS and AQ event routers

Each event is serialized in XML form before being written to the JMS or AQ queue. Each event type can contain different data that is specific to event type. The following is an example of such XML content:

```
<?xml version="1.0" encoding="UTF-8"?>
<CycloneEvent>
    <id>1132766968963.226929@node1</id>
    <type>Messaging.Packaging.SignatureCompleted</type>
    <level>Low</level>
    <created>2005-11-23T12:29:28</created>
    <dispatched>2005-11-23T12:29:28</dispatched>
    <nodeId>node1_te</nodeId>
    <extendedContent/>
    <metaData>
        <MessageSnapshot>
            <CoreId>1132766968539.226918@bdhp6040</CoreId>
            <SenderRoutingId>testsender</SenderRoutingId>
            <ReceiverRoutingId>testreceiver</ReceiverRoutingId>
            <DocumentId>000028161</DocumentId>
            <ContentMimeType>application/EDI-X12</ContentMimeType>
            <MessageState>Actioned</MessageState>
            <MessageContentSize>272</MessageContentSize>
            <BusinessProtocolType>RAW</BusinessProtocolType>
            <BusinessProtocolVersion>1.0</BusinessProtocolVersion>
            <MicCheckFailed>false</MicCheckFailed>
            <IsResponsePositive>false</IsResponsePositive>
            <IsFinalState>false</IsFinalState>
            <MaxRetries>0</MaxRetries>
            <NextRetryNum>0</NextRetryNum>
            <NextRetryTime>1969-12-31T19:00:00</NextRetryTime>
            <CompressionType>None</CompressionType>
            <ReceiptRequested>None</ReceiptRequested>
            <SendAttempt>0</SendAttempt>
            <MaxSendAttempts>0</MaxSendAttempts>
            <ResendInterval>0</ResendInterval>
        </MessageSnapshot>
    </metaData>
</CycloneEvent>
```

Data common to all events

The following data are common to all events written to the JMS or AQ queue.

id

Unique event ID. Each event has its own unique ID.

type

The type of the event.

level

The level of the event (low, high, warning, error).

created

Date and time the event was created in yyyy-MM-dd'T'kk:mm:ss format.

dispatched

Date and time the event was dispatched in yyyy-MM-dd'T'kk:mm:ss format.

nodeid

The ID of the node that created the event.

extendedContent

Extended content associated with the event. Name-values pairs that provide additional data for an event.

Data specific to message-related events

The following data are specific to message-related events written to the JMS or AQ queue.

CoreId

Unique message identifier within the trading engine.

RefToCoreId

Unique message identifier that relates to this message. Typically set on a receipt message, RefToCoreId refers to the request message.

IntegrationId

Used to attach customer-specific ID to a message.

SenderRoutingId

Routing ID of the sending party.

ReceiverRoutingId

Routing ID of the receiving party.

DocumentId

Control ID if the message payload is EDI.

ContentMimeType

The MIME type of the message payload.

MessageState

The state of a message within the trading engine. For example, consumed, produced, rejected.

MessageContentSize

The size of the message content when the event was fired. Note that the message content size changes as the message flows through the trading engine; compression, signing, encryption, packaging affect size at one stage or another.

TransportType

The type of transport that consumed or produced the message. For example, HTTP, HTTPS, FTP, MQSERIES, JMS, SFTP, FILESYSTEM, PLUGGABLE.

PeerAddress

URL of a peer in a peer network.

MessageId

Business protocol message ID.

BusinessProtocolType

Business protocol for the current packaging state. The value depends on a message's state. For example, **raw** represents an unpackaged state and **EDIINT AS1** represents a packaged state for the AS1 protocol.

BusinessProtocolVersion

Version of the business protocol handler for the current packaging state of the message. The value depends on a message's current state. The value indicates the version of the ProtocolSender or ProtocolReceiver handling a message. The value does not necessarily coincide to a specific packaging specification or RFC.

EncryptionAlgorithm

Encryption algorithm used for the message.

DigestAlgorithm

Digest algorithm used for the message.

ContentMic

MIC value of the message.

RejectionDescription

Rejection reason for a rejected message.

ReceivedContentMic

Received content MIC value used when packaging EDIINT MDN. Only set on an outbound MDN message.

MicCheckFailed

True if MIC checking of inbound EDIINT message fails. Only set on an outbound MDN message.

Disposition

Disposition modifier value used when packaging EDIINT MDN. Only set on MDN's that indicate an unpackaging error. Only set on an outbound MDN message.

ResponseCoreId

The core id of the response (acknowledgment) message for this message.

IsResponsePositive

True if the response (acknowledgment) message was positive; **false** otherwise.

IsFinalState

True if the message is in a terminal state (delivered, ignored, rejected, joined, resubmitting).

ebXMLService

For ebXML. identifies the service that acts on the message. The designer defines the service. A service is related actions for an authorized role within a party.

ebXMLAction

For ebXML, identifies a process within a service that processes the message. The action must be unique within the service in which it is defined. The service designer defines the value of the action element.

ebXMLCpaId

For ebXML, identifies the CPA that governs the collaboration between the trading parties. This matches the CPA ID in the CPA, not the name of the CPA XML file.

ebXMLConversationId

For ebXML, a string identifying a set of related messages that comprise a conversation between two parties. It must be unique in the context of the specified CPA ID.

ebXMLRefToMessageId

For ebXML, when present, this must have a MessageId value of an earlier message to which this message relates. If there is no related earlier message, this element must not be used.

ebXMLMessageId

For ebXML, a unique identifier for a message that conforms to RFC 2822.

MaxRetries

Maximum number of times to retry a message after a transport failure.

NextRetryNum

The next retry increment.

NextRetryTime

The time to execute the next retry after a transport failure.

CompressionType

The type of compression used to compress a message.

ReceiptRequested

The type of receipt requested for a message (NONE, SIGNED or UNSIGNED).

ReceiptMicAlgorithm

MIC algorithm used for the receipt.

SendAttempt

The number of sends attempted.

MaxSendAttempts

The maximum number of times to resend a message after a failure to receive a response (acknowledgment).

ResendInterval

The elapsed time to wait between resend attempts.

The alerts.xml file

The alerts.xml file is in [install directory]\[build number]\conf. The default configuration allows certain events in the high, warning and error categories to publish to the user interface. You can edit the file to send the same events by e-mail. This file is only for publishing events for the trading engine.

The default configuration of alerts.xml calls for publishing some but not all of the predefined events. The default configuration reports events in the following general categories:

- Certificates about to expire
- Message errors involving packaging and invalid sender or receiver
- JVM node errors
- Accessing an unlicensed feature
- Transport errors
- Configuration errors
- Unexpected errors

- Fatal errors

If you understand XML you can edit the file to extend reporting to other categories of events.

Send events by e-mail

Use this procedure to configure alerts.xml to have the trading engine send events via e-mail to any recipients. This procedure takes advantage of default settings and requires only minimal changes to alerts.xml to deliver events by e-mail.

Steps

1. Configure a community profile. Make sure to do the following:

   - Configure the global external SMTP server. This is required for e-mailing events. See Configuring external SMTP server on page 24.

   - In the contact information for the community profile, use a valid e-mail address. In some instances, alerts.xml uses this as the “from” address for e-mailed events.

   - Set up a secure e-mail protocol exchange for the community for receiving messages from partners. Use a detached server for this transport; do not use an embedded server. Set up this exchange even though partners might not use it to send messages to your community. In some instances, alerts.xml uses the SMTP delivery address for this exchange as the “from” address for e-mailed events.

2. Make a copy of alerts.xml and keep it as a backup.

3. Open alerts.xml for editing.

4. Near the top of the file, find the **Interval minutes** element. You might want to edit the value. This element limits the number of e-mails that can be sent for the same event. For example, if messages are repeatedly rejected because of an unknown receiver error, only one e-mail message is sent per hour if interval minutes is set to 60.

5. Scroll to the bottom of the file. Here you can find commented-out elements as in Figure 15 for specifying “from” and “to” addresses for e-mailed events.

```
<Communities>
<!-- Uncomment and configure
<Community id="PC1">
<ActionParameters>
<Parameter name="AlertEmailAddresses" value="mail@domain.com"/>
</ActionParameters>
</Community>
-->
<!--
Used whenever there is an event for which there is no community
routing id or there
is no configuration for the community ID for which the event was
generated.
-->
<!-- Uncomment to enable
<Community id="default">
<ActionParameters>
<Parameter name="FromEmailAddress" value="mail@domain.com"/>
<Parameter name="AlertEmailAddresses" value="mail@domain.com"/>
</ActionParameters>
</Community>
-->
</Communities>
```

Figure 15. Address parameters at bottom of alerts.xml

The first block of commented-out elements, identified by **Community id="PC1"**, is for specifying “to” addresses with a specific community as the “from” address. A community routing ID is the value of the **Community id** element. By virtue of the routing ID, the trading engine uses the e-mail address of the secure e-mail protocol exchange for the community as the “from” address. Configuring this block is optional. It is useful if you have two or more community profiles.

The second block of commented-out elements, identified by **Community id="default"**, is for specifying default “from” and “to” addresses for e-mailed events. Configuring default addresses is required even if you use the optional first block. This is because some events are not associated with a community routing ID and the default “to” and “from” addresses must be used for such events.

6 Uncomment the second block of commented-out elements for the default “to” and “from” addresses. Configuring this is required.

For **Parameter name="FromEmailAddress" value**, enter a default “from” address for events.

For **Parameter name="AlertEmailAddresses" value**, enter one or more “to” e-mail addresses. These are where the events will be delivered. To enter two or more addresses, separate each address with a comma. For example, name1@domain.com,name2@domain.com.

7 Optionally, uncomment the first block of commented-out elements for a community-specific “from” address and one or more “to addresses.

Use a community routing ID as the value of **Community id**.

For **Parameter name="AlertEmailAddresses" value**, enter one or more “to” e-mail addresses. These are where the events will be delivered. To enter two or more addresses, separate each address with a comma. For example, name1@domain.com,name2@domain.com.

The trading engine will use the address of the secure e-mail delivery exchange for the community as the “from” address. If such an address is not available, the trading engine will use the contact e-mail in the community profile as the “from” address.

8 Save and close alerts.xml

9 Restart the trading engine server.

When an event defined in alerts.xml is triggered, e-mail messages about the event will be delivered to the specified “to” addresses.

Editing alerts.xml

You can edit the alerts.xml file to alter the events that are published to the database or delivered by e-mail. You must understand XML to change this file. The following describes some of the key elements in the file.

Interval minutes

The trading engine only fires one action (e-mail or database) for multiple similar events that occur within a specified interval. If the following meta-data matches, the events are considered duplicates. This is a way to limit the number of e-mail messages sent for the same events in a given period of time.

AlertDefinition.Name
Event.EventType
Event.EventLevel
Event.ExtendedEventContent
Message.SenderRoutingId
Message.ReceiverRoutingId
Message.BusinessProtocolType
Message.BusinessProtocolVersion

AlertDefinition

AlertDefinition elements define the type of events to publish, where to deliver the events (database or e-mail) and the information to include in event messages.

Events

Events elements refer to some of the predefined event types.

Actions

One or more actions can be taken when an alert definition fires. Two actions are supported: **database** and **email**. the database action delivers triggered events to the user interface. The email action delivers events by e-mail.

Address

Each email Action type element has an Address parameter. The default value of **{AlertEmailAddressesses}** means the “to” addresses in the Communities elements at the bottom of the file are used. You can substitute addresses without brackets for the bracketed parameter value.

Format

Format elements control the format and content of the e-mail messages. Valid formats are **text** and **XML**. If the format is **text**, a list of Field elements can be defined that supply the content of the e-mail message. Each Field element results in a new line in the e-mail message. The field has a label and a hard-coded value or a reference to some meta-data contained in the event. See Meta-data for e-mail events on page 110

Communities

The Communities element defines the values for any bracketed Address parameters referenced in the Actions element. For every event, if there is a community ID and the action is email, then the name of the bracketed parameter is looked up for that community ID. This allows the e-mail address to be configured for each community.

Meta-data for e-mail events

The following meta-data are available to all events in alerts.xml.

Table 6 - Meta-data for e-mail events

Meta-data	Description
AlertDefinition.Name	Name of the alert definition
Event.Name	Name of event
Event.EventType	Type of event (usually same as Event.Name)
Event.EventLevel	Level of event (high, warning, error)
Event.Timestamp	Time event was triggered.
Event.ExtendedEventContent	Extra information that may be with the event
Event.NumberOfOccurances	Number of times the event has been published

The following meta-data are available to events at the Messaging level (see Messaging on page 113) in the default configuration of alerts.xml. All of these fields names are included in generated e-mail messages. Fields without data display as blank in e-mail messages.

Message.CoreId
Message.RefToCoreId
Message.SenderRoutingId
Message.ReceiverRoutingId
Message.DocumentId
Message.MimeType
Message.MessageState
Message.MessageContentSize
Message.TransportType
Message.PeerAddress
Message.MessageId
Message.BusinessProtocolType
Message.BusinessProtocolVersion
Message.EncryptionAlgorithm
Message.DigestAlgorithm
Message.ContentMic
Message.RejectedReason
Message.ReceivedContentMic
Message.MicCheckFailed
Message.Disposition
Message.SenderRoutingIdType
Message.ReceiverRoutingIdType
Message.TrueSenderRoutingIdType
Message.TrueSenderRoutingId
Message.TrueReceiverRoutingIdType
Message.TrueReceiverRoutingId

Message.PeerAddress
Message.ResponseCoreId
Message.IsResponsePositive
Message.IsFinalState
Message.EbxmlService
Message.EbxmlAction
Message.EbxmlCpaId
Message.EbxmlConversationId
Message.EbxmlRefToMessageId
Message.EbxmlMessageId
Message.MaxRetries
Message.NextRetryNum
Message.NextRetryTime
Message.CompressionType
Message.ReceiptTypeRequested
Message.SendAttempt
Message.MaxSendAttempts
Message.ResendInterval

Event tables

Tables of all events are provided in the following topics.

Events follow a hierarchy that lets you filter the volume of reported events. For example, the following event from the table Messaging on page 113 is the full detail for this event:

Messaging.Message.MessageUnpackaged.Request

Using an event filter, you can specify how many Messaging events are published. If you filter events at the top level, **Messaging**, all events at that level and below are reported. This would include all events in the following tables: Messaging on page 113, Transport on page 120 and Packaging on page 121.

You could exclude many events by filtering according to the next level in the hierarchy. For example, if you filtered at the **Messaging.Message** level, all events in the table Messaging would be reported. No events would be reported from the tables Transport and Packaging.

You could drop another level and exclude even more events. For example, if you filtered at the **Messaging.Message.MessageUnpackaged** level, only three events from the table Messaging would pass the filter.

The tables give the level of each event.

A “level” indicates the significance of an event. Levels rank events from low to high importance. There are four levels, which are listed in order of lowest to highest importance: Low, High, Warning and Error.

Event levels provide different degrees of information. Low level events contain the most detailed information and high level events contain the most general information. The warning and error levels do not correspond to a level of detail, but do infer degrees of importance.

Messaging

The following are events that occur during business message processing.

Messaging.Message. is the prefix for these events. In other words, the syntax of the first event in the following table is actually **Messaging.Message.ActionTreeExecuted**.

Table 7 - Messaging events. Prefix: Messaging.Message.

Event	Lvl	Description
ActionTreeExecuted	Low	Action tree executed
Collaboration.ErrorBuildingCollaboration	Error	Error building a binary collaboration
Collaboration.ErrorExecutingActionTree	Error	Error executing a binary collaboration
DecompressionFailure	Error	Message could not be decompressed
DecryptionFailure	Error	Message could not be decrypted
Duplicate	Warn	Received a duplicate message
Duplicate.Message	Warn	Received a duplicate message
Duplicate.Payload	Error	Received a duplicate document

Table 7 - Messaging events. Prefix: Messaging.Message.

Event	Lvl	Description
InlineProcessing.Error	Error	Inline processing was performed on a message that resulted in an exception being thrown
InlineProcessing.Initiated	Low	This event generates before an inline process is called by a message processing action or a delivery exchange
InlineProcessing.Performed	Low	Inline processing was applied to a message and returned successfully
InvalidCertificate	Error	No valid certificate
InvalidCertificate.InvalidEncryptionCertificate	Error	No valid encryption certificate
InvalidCertificate.InvalidSigningCertificate	Error	No valid signing certificate
InvalidPartner	Error	Partner ID invalid or unknown
InvalidPayload	Error	Payload cannot be recognized
InvalidPayload.XmlParsingFailure	Error	Error parsing XML payload
InvalidResponseRequest	Error	Message contains an invalid receipt request
InvalidResponseRequest.UnsupportedFormat	Error	Requested protocol format is not supported
InvalidResponseRequest.UnsupportedMicAlgorithms	Error	Requested MIC algorithms are not supported

Table 7 - Messaging events. Prefix: Messaging.Message.

Event	Lvl	Description
InvalidSignature	Error	Signature cannot be verified or validated
InvalidSignature.InvalidHash	Error	Signature has invalid hash value
InvalidSignature.UnassociatedCertificate	Error	Signature certificate is not associated with the message sender
InvalidSignature.UntrustedCertificate	Error	Signature certificate is untrusted
MessageIgnored	High	
MessageJoined	High	For messages with multiple payloads (typically ebXML and RosettaNet), child messages of the original have been joined into the single message sent
MessagePackaged	Low	Message packaged
MessagePackaged.Error	Error	An error was encountered while packaging a message
MessagePackaged.Request	Low	A request was packaged
MessagePackaged.Response	Low	A receipt was packaged
MessageReceived	High	Message received from transport
MessageRejected	Error	Message rejected
MessageSent	High	Message sent via transport
MessageUnpackaged	Low	Message unpackaged

Table 7 - Messaging events. Prefix: Messaging.Message.

Event	Lvl	Description
MessageUnpackaged.Error	Error	An error was encountered while unpackaging a message
MessageUnpackaged.Request	Low	A document was unpackaged
MessageUnpackaged.Response	Low	A receipt was unpackaged
PayloadDelivered	High	Payload delivered to integration point
PayloadReceived	High	Payload received from integration point
PayloadSplit	High	A batch EDI document has been split into individual documents
PayloadSplit.Error	Error	An unrecoverable error occurred when splitting an EDI document. No children were produced and the original document failed.
RequiredHeaderNotFound	Error	Message did not contain the required header
ResendCancelled	Low	Resend cancelled
ResendInitiated	High	Resend initiated
ResendsExhausted	Error	Message resent maximum times

Table 7 - Messaging events. Prefix: Messaging.Message.

Event	Lvl	Description
ResponseDelivered	High	An inbound receipt has been reconciled with the outbound message that requested it, or an outbound receipt has been successfully transmitted to the requesting party.
ResponseSent	High	Receipt sent via transport
Resubmit	High	Message resubmitted
Resubmit.Failed	Error	Message failed to resubmit
Resubmit.Initiated	High	Message resubmission initiated
UnexpectedProcessingError	Error	Cannot process a received message due to an unexpected error
UnexpectedResponse	Error	Receipt message is unexpected or contain
UnexpectedResponse.AuthenticationFailed	Error	Receipt indicates authentication of original message failed
UnexpectedResponse.DecompressionFailed	Error	Receipt indicates decompression of original message failed
UnexpectedResponse.DecryptionFailed	Error	Receipt indicates decryption of original message failed

Table 7 - Messaging events. Prefix: Messaging.Message.

Event	Lvl	Description
UnexpectedResponse.DuplicateRequest	Warn	Receipt indicates the original message was a duplicate
UnexpectedResponse.DuplicateResponse	Warn	Received a duplicate receipt
UnexpectedResponse.InsufficientMessageSecurity	Error	Receipt indicates original message had insufficient security
UnexpectedResponse.IntegrityCheckFailed	Error	Receipt indicates integrity check of original message failed
UnexpectedResponse.MicMismatch	Error	MIC in receipt does not match MIC in original message
UnexpectedResponse.NoPartner	Error	Receipt indicates there was no partner configured for the original message
UnexpectedResponse.SenderEqualsReceiver	Error	Receipt indicates the sender and receiver of the original message were identical
UnexpectedResponse.UnexpectedProcessingError	Error	Receipt indicates there was an unexpected error processing the original message
UnexpectedResponse.UnknownError	Error	Received a receipt with an unknown error
UnexpectedResponse.UnknownFailure	Error	Received a receipt with an unknown failure

Table 7 - Messaging events. Prefix: Messaging.Message.

Event	Lvl	Description
UnexpectedResponse.UnknownWarning	Warn	Received a receipt with an unknown warning
UnexpectedResponse.UnmatchedResponse	Error	Receipt does not match original message
UnexpectedResponse.UnsupportedFormat	Error	Receipt indicates the original message requested a receipt using an unsupported protocol format
UnexpectedResponse.UnsupportedMicAlgorithms	Error	Receipt indicates the original message requested a receipt with an unsupported MIC algorithm
UnknownReceiver	Error	Received message from unknown receiver
UnknownSender	Error	Received message from unknown sender
ValidationFailure	Error	An error validating an ebXML message. For instance, a message could not be validated against a CPA.
ValidationFailure.NotEncrypted	Error	An inbound message failed because it was not signed
ValidationFailure.NotSigned	Error	An inbound message failed because it was not encrypted

Transport

The following are transport events.

Messaging.Transport. is the prefix for these events. In other words, the syntax of the first event in the following table is actually **Messaging.Transport.ConnectionClosed**.

Table 8 - Transport events. Prefix: Messaging.Transport.

Event	Lvl	Description
ConnectionClosed	Low	Input or output connected closed
ConnectionEstablished	Low	Outbound or inbound connection established
ConnectionInitiated	Low	Outbound or inbound connection initiated
InputTransferCompleted	Low	Input transfer has completed
InputTransferInitiated	Low	Input transfer initiated on new connection
OutputTransferCompleted	Low	Output transfer completed
OutputTransferInitiated	Low	Output transfer initiated
PostProcessing	Low	Post processing
PostProcessing.Completed	High	Post processing completed
PostProcessing.Failure	Error	Post processing failed
PostProcessing.Initiated	High	Post processing started
ReceiveFailure	Error	A receive (input) fails for any reason

Table 8 - Transport events. Prefix: Messaging.Transport.

Event	Lvl	Description
ReceiveFailure.InputConnectionFailure	Error	Input connection cannot be set up
ReceiveFailure.InputConnectionLost	Error	Input connection lost
ReceiveFailure.InputTermination	Error	Input abnormally terminated
RetryCompleted	Low	Retry finished
RetryFailure	Error	A retry fails for any reason
RetryFailure.RetriesExhausted	Error	All retries have been attempted
RetryFailure.RetryCancelled	Error	Retry operation cancelled
RetryScheduled	Low	A message that failed to send due to a transport error has been scheduled for another attempt at sending
SendFailure	Error	A send (output) fails for any reason
SendFailure.OutputConnectionFailure	Error	A connection cannot be established
SendFailure.OutputConnectionLost	Error	Established connection lost
SendFailure.OutputTermination	Error	Output abnormally terminated

Packaging

The following are message packaging events.

Messaging.Packaging. is the prefix for these events. In other words, the syntax of the first event in the following table is actually **Messaging.Packaging.CertificateValidationFailure**.

Table 9 - Packaging events. Prefix: Messaging.Packaging.

Event	Lvl	Description
CertificateValidationFailure	Error	Certificate validation failed
CompressionCompleted	Low	Compression completed
CompressionInitiated	Low	Compression started
DecompressionCompleted	Low	Decompression completed
DecompressionFailure	Error	Decompression failed
DecompressionInitiated	Low	Decompression started
DecryptionCompleted	Low	Decryption completed
DecryptionFailure	Error	Decryption failed
DecryptionInitiated	Low	Decryption started
EncryptionCompleted	Low	Encryption completed
EncryptionInitiated	Low	Encryption started
PackageCompleted	Low	Packaging completed
PackageInitiated	Low	Packaging started
PayloadUnpackaged	Low	A payload has been unpackaged
SignatureCompleted	Low	A signing operation has completed
SignatureInitiated	Low	A signing operation has started
SignatureVerificationCompleted	Low	A signature verification and validation has completed
SignatureVerificationFailure	Error	Signature verification failed

Table 9 - Packaging events. Prefix: Messaging.Packaging.

Event	Lvl	Description
SignatureVerificationInitiated	Low	A signature verification and validation has started
UnpackageCompleted	Low	Unpackaging completed
UnpackageInitiated	Low	Unpackaging started

Peer network

The following are peer network events.

Messaging.PeerNetwork. is the prefix for these events. In other words, the syntax of the first event in the following table is actually **Messaging.PeerNetwork.ReceiptForwarded**.

Table 10 - Peer network events. Prefix: Messaging.PeerNetwork.

Event	Lvl	Description
ReceiptForwarded	Low	A peer received a receipt intended for another peer and forwarded the receipt to the proper peer.
ReceiveFailed	Error	An error occurred while processing a peer message from another peer.

Administration alert

The following is an event that occurs when an alert is generated. Such an event also contains information about the particular alert. See The alerts.xml file on page 106.

Table 11 - Administration alert.

Event	Lvl	Description
Administration.Alert	High	An alert generated

Administration configuration

The following are administration configuration events, which track changes to the system and resources used and accessed.

Administration.Configuration. is the prefix for these events. In other words, the syntax of the first event in the following table is actually **Administration.Configuration.CertificateManagement.PersonalCertificateAboutToExpire**.

Table 12 - Administration events. Prefix: Administration.Configuration.

Event	Lvl	Description
CertificateManagement.PersonalCertificateAboutToExpire	High	A certificate is about to pass its expiration date
CertificateManagement.PersonalCertificateAdded	High	A certificate has been added
CertificateManagement.PersonalCertificateExpired	High	A certificate has expired
CertificateManagement.PersonalCertificateNotOperational	High	A certificate is not in active status
CertificateManagement.PersonalCertificateNotYetValid	High	A certificate precedes its validation date
CertificateManagement.PersonalCertificateRemoved	High	A certificate has been removed
CertificateManagement.PersonalCertificateRevoked	High	A certificate is no longer valid
CertificateManagement.PsePolicyUpdated	High	A default signing or encryption certificate has been changed
CertificateManagement.SslCertificateAboutToExpire	High	An SSL certificate is about to pass its expiration date
CertificateManagement.SslCertificateExpired	High	An SSL certificate has expired
CertificateManagement.SslCertificateNotOperational	High	An SSL certificate is not in active status

Table 12 - Administration events. Prefix: Administration.Configuration.

Event	Lvl	Description
CertificateManagement.SslCertificateNotYetValid	High	An SSL certificate precedes its validation date
CertificateManagement.SslCertificateRevoked	High	An SSL certificate is no longer valid
CertificateManagement.TrustedCertificateAboutToExpire	High	A trusted certificate is about to pass its expiration date
CertificateManagement.TrustedCertificateAdded	High	A certificate has been trusted
CertificateManagement.TrustedCertificateExpired	High	A trusted certificate has expired
CertificateManagement.TrustedCertificateNotOperational	High	A trusted certificate is not in active status
CertificateManagement.TrustedCertificateNotYetValid	High	A trusted certificate precedes its validation date
CertificateManagement.TrustedCertificateRemoved	High	A certificate has been untrusted
CertificateManagement.TrustedCertificateRevoked	High	A trusted certificate is no longer valid
CertificateManagement.UserCertificateAboutToExpire	High	A user’s certificate is about to expire
CertificateManagement.UserCertificateExpired	High	A user’s certificate has expired
CertificateManagement.UserCertificateNotOperational	High	A user’s certificate is not in active status
CertificateManagement.UserCertificateNotYetValid	High	A user’s certificate precedes its validation date
CertificateManagement.UserCertificateRevoked	High	A user’s certificate is no longer valid
Collaboration.AllBuildingBlocksRemoved	High	A message handler condition has been removed

Table 12 - Administration events. Prefix: Administration.Configuration.

Event	Lvl	Description
Collaboration.BuildingBlockAdded	High	A message handler condition has been added
Collaboration.BuildingBlockRemoved	High	A message handler condition has been removed
CrlManagement.UpdateCompleted	High	An update of a certificate revocation list has been completed
CrlManagement.UpdateFailed	Error	An update of a certificate revocation list has failed
CrlManagement.UpdateInitiated	Low	An update has been initiated for a certificate revocation list
EmbeddedServer.Added	High	An embedded server has been added
EmbeddedServer.Removed	High	An embedded server has been deleted
EmbeddedServer.Updated	High	An embedded server has been updated
ExchangePoint.Added	High	An exchange point has been added
ExchangePoint.Removed	High	An exchange point has been deleted
ExchangePoint.Updated	High	An exchange point has been updated
Node.Added	High	A node has been added
Node.Removed	High	A node has been deleted

Table 12 - Administration events. Prefix: Administration.Configuration.

Event	Lvl	Description
Party.CommunityAdded	High	A community has been added
Party.CommunityImported	High	A community profile has been imported
Party.CommunityRemoved	High	A community has been deleted
Party.CommunityUpdated	High	A community has been updated
Party.ImportFailed	Error	A community or partner profile failed to import
Party.PartnerAdded	High	A partner has been added
Party.PartnerImported	High	A partner profile has been imported
Party.PartnerRemoved	High	A partner has been deleted
Party.PartnerSelfRegistered	High	A web trader partner has registered and awaits sponsor approval
Party.PartnerUpdated	High	A partner has been updated

CSOS

The following are events related to CSOS document processing. These events are generated only if your user license allows CSOS processing.

Table 13 - CSOS events.

Event	Lvl	Description
CSOS.Messaging.Approval	Low	A user has signed and approved an outbound message

Table 13 - CSOS events.

Event	Lvl	Description
CSOS.Messaging.Approval.Rejected	High	A message awaiting signing was rejected
CSOS.Messaging.QueuedForApproval	Low	A document is waiting for user signing
CSOS.Messaging.Verification	Low	The user signature of an inbound message has been verified
CSOS.Messaging.Verification.Failed	Error	The user signature of an inbound message has failed verification

Terminal events

The significance of the following events, and the reason for presenting them apart from other events, is these are terminal events, which indicate the end of processing in the trading engine.

Messaging.Message.PayloadDelivered

Level: High

For an inbound message, the payload was sent to integration.

For an outbound message, the payload was sent to the partner and a receipt (if expected) was received.

Messaging.Message.ResponseDelivered

Level: High

For an inbound receipt, it has been received and processed successfully,

For an outbound receipt, it has been sent successfully.

Messaging.Message.MessageRejected

Level: Error

An inbound or outbound message containing a payload was rejected.

This could due to a transport or security failure, application configuration error, or systems error. The reason for the rejection and other information are provided with the response.

Messaging.Message.MessageIgnored

Level: High

Processing has reached a point where the message can be safely ignored and not processed further. The following are the known instances of when a message is ignored.

When an AS2 message containing a payload or response is sent, the received synchronous HTTP response is treated as a message. If a synchronous MDN is not expected and the HTTP response is a 200-level response, the message is ignored.

When the sender for the raw business protocol is asked to send a message that does not contain any content (either there really is no content or the content is of zero length), the message is ignored.

When the web services protocol receiver receives a duplicate of a previously received message, the duplicate is ignored.

When the ebXML protocol receiver receives an asynchronous SOAP fault, the SOAP fault message is ignored.

When the ebXML protocol receiver receives a synchronous SOAP fault for an unknown message, the SOAP fault message is ignored.

When the ebXML protocol receiver receives a duplicate of a previously received message, all the messages containing the payloads of the duplicate message are ignored.

Administration system

Administration.System. is the prefix for these events. In other words, the syntax of the first event in the following table is actually **Administration.System.ClusterBus.ErrorEvent**.

Table 14 - Administration system events. Prefix: Administration.System.

Event	Lvl	Description
ClusterBus.ErrorEvent	Error	A node communication error has occurred
ClusterBus.FatalEvent	Error	A node communication fatal error has occurred
ClusterBus.WarningEvent	Warn	A node communication warning has occurred
Node.ShutdownCompleted	High	Node has shut down
Node.ShutdownInitiated	High	Node begins to shut down
Node.StartupCompleted	High	Node has started
Node.StartupFailure	Error	Node fails to start
Node.StartupInitiated	High	Node begins to start up
StartupCompleted	High	Server has started
StartupInitiated	High	Server begins to start up
SystemLoadExceeded	Warn	The system has exceeded the processing threshold and has throttled back

Administration licensing

Table 15 - Administration licensing event

Event	Lvl	Description
Administration.Licensing.ResourceUnlicensed	Warn	License.xml file does not allow access to a resource

Administration persistence

Table 16 - Administration persistence event

Event	Lvl	Description
Administration.Persistence.MessagesPurged	High	Messages have been purged from the backup directory and the database

Security

The following events are related to user-level security auditing.

Security. is the prefix for these events. In other words, the syntax of the first event in the following table is actually **Security.Authentication.AuthenticationFailure**.

Table 17 - Security events. Prefix: Security.

Event	Lvl	Description
Authentication.AuthenticationFailure	High	A user cannot be authenticated
Authentication.AuthenticationSuccess	High	A user has been authenticated
Authentication.LockedOutUserAttempt	High	A user retrys logging on
Authentication.UserLoggedOut	High	A user has logged out
Authentication.MaxAttemptsExceeded	High	A user has exhausted the number of allowed log-on attempts
Authorization.AuthorizationDenied	High	A user has been denied use of a resource
Authorization.AuthorizationGranted	Low	A user has been authorized to use a resource

Table 17 - Security events. Prefix: Security.

Event	Lvl	Description
Configuration.Failure	Error	Error due to missing or corrupted data in the database or a missing license.xml file
Configuration.ItemChanged	High	A security configuration has been changed
Configuration.PasswordCreationFailure	Error	A password could not be encrypted
Group.GroupAdded	High	A role has been added
Group.GroupRemoved	High	A role has been deleted
Group.GroupUpdated	High	A role has been updated
Group.UserAddedToGroup	High	A user has been added to a role
Group.UserRemovedFromGroup	High	User removed from a role
User.LockedOut	High	A user’s authority to log on has been suspended
User.LockOutRemoved	High	A user can log on again
User.PasswordUpdated	High	A user’s password has been changed
User.UserAdded	High	A user has been added
User.UserRemoved	High	A user has been deleted
User.UserUpdated	High	A user has been updated

❖ ❖ ❖

10 Getting started

Cyclone Activator, BEA WebLogic Edition provides a secure, scalable gateway for B2B collaboration. The unified framework helps establish relationships with trading partners, transact business over the Internet, and integrate with back-end systems. The gateway provides flexibility in connecting to partners and legacy systems using widely used protocols, transports, and integration methods.

The application's user interface integrates gateway management, monitoring and metrics into one view.

AS1, AS2, AS3, RosettaNet, and ebXML are among the protocols Cyclone Activator supports for exchanging messages with partners. User-defined message routing and rules-based processing can be handled at the partner or document level.

Cyclone Activator supports multiple operating systems for ease of deployment.

The following topics provide basic information about the application and guidelines for configuration.

Concepts

How the trading engine processes

Figure 16 presents a high-level view of how the trading engine processes outbound and inbound documents. This shows typical document flows, although your organization’s configuration may differ.

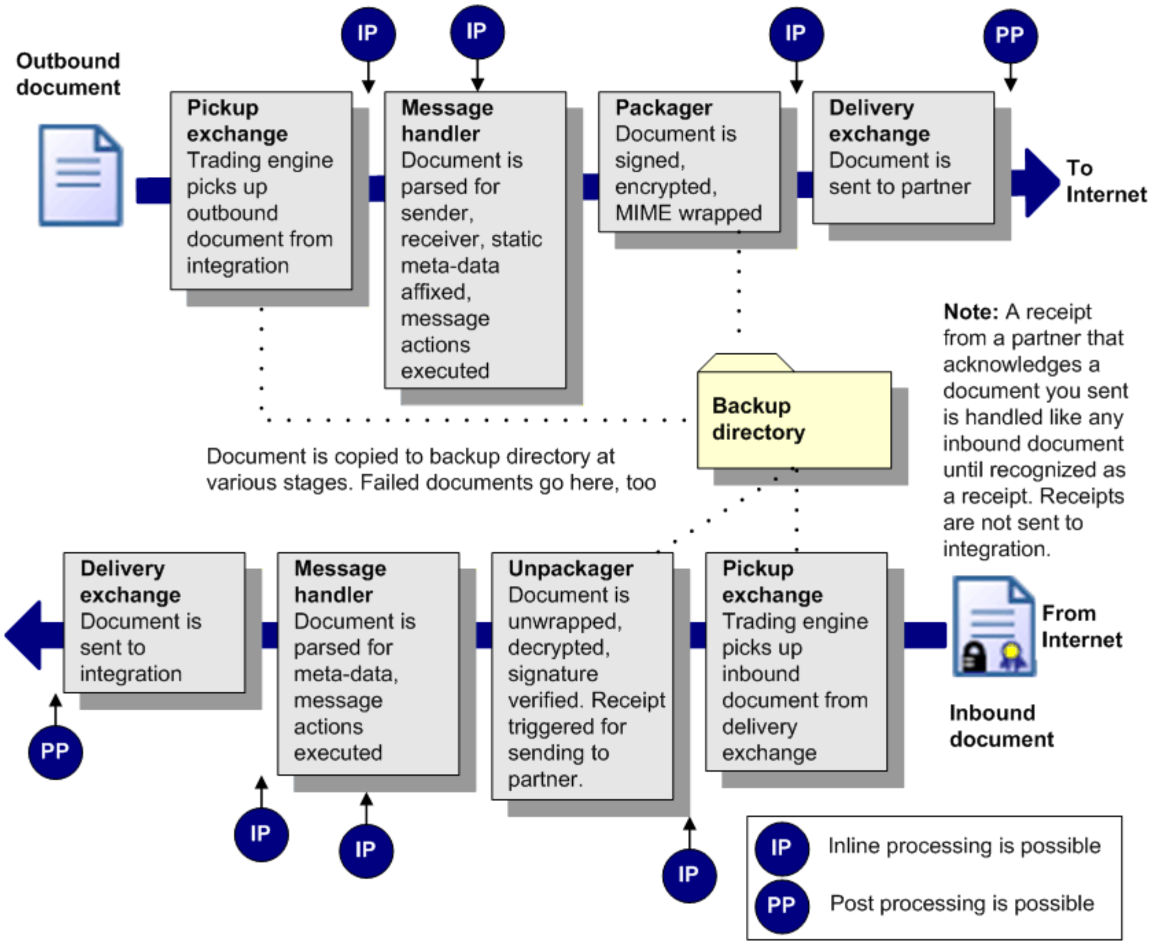

Outbound document
IP
IP
IP
PP
Pickup exchange
Trading engine picks up outbound document from integration
Message handler
Document is parsed for sender, receiver, static meta-data affixed, message actions executed
Packager
Document is signed, encrypted, MIME wrapped
Delivery exchange
Document is sent to partner
To Internet
Note: A receipt from a partner that acknowledges a document you sent is handled like any inbound document until recognized as a receipt. Receipts are not sent to integration.
Backup directory
Document is copied to backup directory at various stages. Failed documents go here, too
Delivery exchange
Document is sent to integration
Message handler
Document is parsed for meta-data, message actions executed
Unpackager
Document is unwrapped, decrypted, signature verified. Receipt triggered for sending to partner.
Pickup exchange
Trading engine picks up inbound document from delivery exchange
From Internet
Inbound document
PP
IP
IP
IP
IP Inline processing is possible
PP Post processing is possible

Figure 16. Outbound and inbound processing flow

Major components

The following describes the system's major components.

Administrator is a browser-based application that enables you to configure and maintain your system for document exchanges. The parameters you set using the Administrator application are stored in the system database.

Administrator enables you to monitor document traffic. You can search for documents and re-submit documents to partners or for integration.

Server performs the document transfers. Server reads the parameters from the database and uses them to process, send and receive documents over the Internet. Server is designed for continuous operation, 24 hours a day, seven days a week.

Trading engine glossary

The following define common terms used in describing configuration or operation of the trading engine.

Term	Description
certificate	A digital certificate contains keys used for encrypting and signing messages, and also for decrypting and verifying signatures. A certificate can contain a public-private key pair or a public key only. See **key**.
collaboration settings	Control packaging for messages a community sends to partners. Determines whether messages are sent in cipher or plain text, and whether partners must acknowledge receiving messages.
community profile	A community profile represents your local way of grouping trading partners. It defines your organization’s internal processes for handling messages. It also defines how your community expects to receive messages from partners. In versions 4.2 and earlier, a community profile was called a company profile.
delivery exchange	The message protocol or transport or both a community or partner uses to send and receive messages.
document	A business document such as a purchase order, invoice, shipping notice, and so on.
EDI	Electronic data interchange.
EDIFACT	Electronic Data Interchange For Administration Commerce and Transport. An EDI standard of the International Organization for Standardization (ISO).

Term	Description
inbound	A message received over the Internet from a partner. An inbound message is unpacked and sent to inbound integration.
integration	The transport to the back-end system for picking up outbound messages or delivering inbound messages. Supported integration transports include file system, FTP, JMS, MQSeries.
key	Keys are contained in digital certificates. There are two kinds of keys: Private and public. A private key is your secret key for decrypting messages or signing messages. A public key is also your key, but it can be used by a partner to encrypt messages that only you can decipher with your private key.
message	The cargo or payload the trading engine handles. A message can be a business document or a receipt that you or a partner send to acknowledge receiving a document.
message protocol	A standard or convention for handling the exchange of e-commerce business messages over the Internet, and sometimes between back-end systems and the trading engine. A message protocol supports one or more transports. Supported message protocols include AS1, AS2, AS3, ebXML, RNIF.
message validation	Rules stating whether a community accepts messages from partners that are encrypted or plain text and signed or unsigned. In the case of EDI documents, a community also can specify whether to accept duplicate messages.
outbound	A message sent over the Internet to a partner. The trading engine picks up the message from outbound integration and packages it before sending.
packaging	How an outbound message is prepared before it is sent to a partner. Packaging normally involves signing, encrypting and MIME-wrapping the payload.
partner profile	A partner profile specifies the message protocol and transport for sending messages over the Internet to the partner.

Term	Description
payload	The business document within a packaged message.
receipt	A transport-level message that you or a partner send to acknowledge receiving a document.
resend	After waiting for the partner to send a receipt acknowledging receiving a message, the trading engine sends the message again on the presumption the partner did not receive the earlier message. This sometimes is confused with retry.
retry	A subsequent attempt to connect to the partner's transport when the initial attempt to connect and send the message failed. This sometimes is confused with resend.
TRADACOMS	TRAding DAta COMmunicationS. An EDI standard of the Article Numbering Association. Widely used in the United Kingdom.
trading	The exchange of e-commerce messages over the Internet between trading partners.
transport	The protocol for moving messages between the trading engine and partners over the Internet or a back-end system. Supported transports include, SMTP, HTTP, FTP, JMS, file system.
unpackaging	The action of unwrapping an inbound packaged message.
X12	An EDI protocol of the American National Standards Institute. It is the primary North American standard for defining EDI transactions.

Configuration outline

Use the following outline as a guide for configuring Cyclone Activator. This outline presumes the application has been installed, the server has been started, and you have logged on to the user interface in a browser. If not, first see Installation on page 11.

1 In the user interface, set up your community profile. The user interface guides you through much of this process.

A community profile contains information about your organization and its preferences for exchanging messages with other partners.

See Trading configuration on page 141 for information about community and partner profiles and the roles they play in Cyclone Activator. See Add a community on page 146 for how to add a community profile in the user interface.

2. Generate or obtain a certificate for your community profile. The trading engine uses certificates for signing and encrypting messages.

   See Manage certificates on page 325.

3. Determine whether you want to complete any other configuration for your community profile.

   See Check list for community configuration on page 146.

4. Determine whether messages sent or received by you or partners must pass through proxy servers, firewalls or load balancers. Take steps to adjust the trading engine configuration to accommodate unhindered message traffic. See Firewalls and proxy servers on page 155.

5. Export your community profile and distribute it by a secure means to your trading partners. See Export a community profile on page 151.

   A link for exporting the profile to a file is provided at the bottom of the community summary page in the user interface. If you send the profile file to partners as an e-mail attachment, we recommend compressing it with WinZip or similar software to ensure file integrity.

6. Import or create profiles for your trading partners. The user interface guides you through most of this process.

   See The partner profile on page 148 and Add a partner on page 150.

7. Perform test trading to validate your configuration.

   See Test trading on page 433.

8. Begin trading business documents with your partner.

   When setting up your community, you configure how the trading engine retrieves documents to be sent to partners and how documents received from partners are delivered. Meanwhile, the partner profile specifies how documents are to be transported to partners. With the server running and with all required configuration in place, trading should take place, presuming your partner also has completed all configuration tasks.

The trading engine retrieves documents from the pickup integration delivery exchange specified in the community profile. It packages documents according to the community collaboration settings and sends documents according to the default delivery exchange configured in the partner profile. Documents the community receives from partners are routed to the integration delivery exchange specified in the community profile.

9. Use the Message tracker area of the user interface to view records of trading activity. See Message tracker on page 411.

Interoperability

The trading engine has been certified for interoperability for AS1, AS2, AS3 and ebXML. See www.ebusinessready.org for a list of software that the trading engine has been successfully tested with for interoperability.

Security guidelines

To ensure the integrity of the data processed, we recommend the following security measures in addition to your company's own security policies. Although the risks are possibly remote, failure to institute minimum security measures may result in compromised data.

1. Install the trading engine in the data layer behind a firewall and not in an area unprotected from exposure to the Internet.
2. Do not install or run the trading engine under a privileged account. This includes root in UNIX and administrator or system accounts in Windows.
3. Do not view a binary document in the user interface that has been received by the trading engine without first scanning the document for viruses.
4. Institute a policy for periodically changing the passwords for accessing the user interface.

5. Control access to the computer running the trading engine to authorized users.

6. When distributing your certificate to partners, do so via a secure means. Encourage your partners to do likewise.

Maintenance tips

Do the following to maintain the trading engine and its data:

- Back up all system directories and files as part of your normal backup schedule.
- Review the system logs at frequent intervals to detect potential problems. See Log file tracking on page 82.
- Delete old log files from the following locations: [install directory]\common\logs and [install directory]\[build number]\webapps\ui\WEB-INF\logs.
- Set up a schedule for purging database records and the corresponding file backups. See Data backups and deletes on page 421.
- Make sure there is enough disk space available for the system and the documents you exchange.
- Use your available system tools to check memory usage.

11 Trading configuration

Setting up a trading configuration involves adding a community profile, which contains information about an organization and how it wants to receive messages from partners. Completing the configuration involves adding one or more partners to the community. A partner profile contains information about a partner and how to send messages to it. The user interface provides helpful links and prompts for adding communities and partners.

Concepts

Procedure

How profiles work

The trading engine organizes the information needed to exchange documents with partners in community and partner profiles. The use of profiles makes it easy to set up and maintain trading relationships. A community profile defines how you receive documents from partners. A partner profile defines how you send documents to a partner. The user interface guides you through most steps for setting up community and partner profiles.

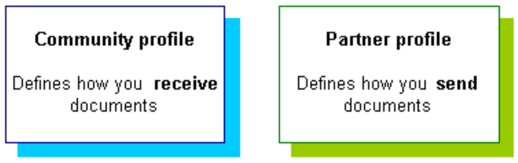

Community profile
Defines how you receive documents
Partner profile
Defines how you send documents

Figure 17. Purpose of community and partner profiles

If you and your partner use Cyclone Commerce software, you can take advantage of the application's profile management features, such as profile and certificate exporting and importing. If a partner uses other interoperable software, you must maintain the partner profile manually.

To establish a trading relationship, you create and export your community profile to your trading partner, who imports it as your partner profile. Conversely, your partner creates and exports a community profile that you import as a partner profile on your system.

Figure 18 shows the community-partner profile exchange within the trading engine. This example uses Worldwide Trading as the local trading partner and Acme Industries as the remote trading partner.

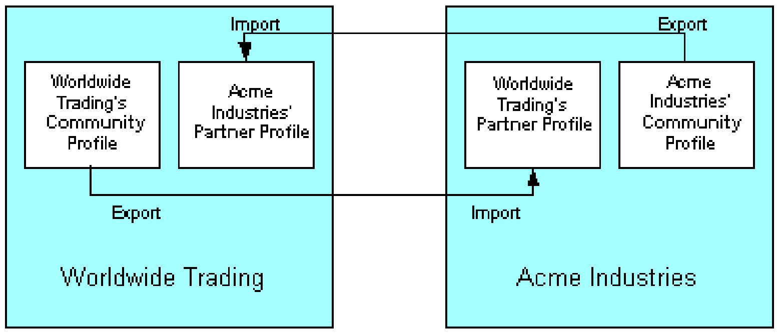

Import
Export
Worldwide Trading's Community Profile
Acme Industries' Partner Profile
Worldwide Trading's Partner Profile
Acme Industries' Community Profile
Export
Import
Worldwide Trading
Acme Industries

Figure 18. Example of community-partner profile exchange

The community profile

A community represents your local way of grouping trading partners. It defines your organization's internal processes for handling messages. It also defines how your community expects to receive messages from partners.

The local information is used by your system to set document back-up options, tune system performance and integrate with back-end systems. While these settings are significant to you, they are not relevant for your partners.

By contrast, the trading information in the community profile is important to your partners. It consists of what message protocols and transports you want partners to use when sending documents to you. If you want to securely exchange documents, the exported community profile also contains a copy of your certificate and public key used for encryption.

In versions of Cyclone Activator earlier than 5.0, community profiles are called company profiles.

Significant elements of a community profile are:

- The profile name.
- A routing ID partners can use to send you documents. A community can have multiple routing IDs.
- An integration pickup delivery exchange for receiving documents from a back-end system for packaging and sending to partners. (The transport for actually sending documents is specified in the partner profile and not the community profile.)
- One or more delivery exchanges that you support for receiving documents from partners over the Internet.
- An integration delivery exchange for sending documents received from partners to a back-end system.
- For secure trading, a certificate with a public-private key pair. Your private key remains in your system. Only the public key and certificate are provided to partners in the exported community profile.

Anatomy of a community profile

The user interface displays a labeled graphic that represents the parts of a community profile. The graphic (Figure 19) appears at the top of the community summary page and other community-related pages. Place your cursor over a label and a red box appears. Click inside the red box to go to the page the label references.

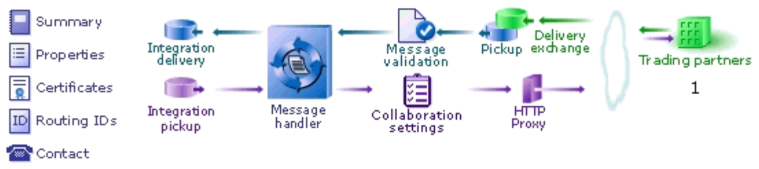


Figure 19. Community navigation graphic

The following defines the role of each labeled part of the navigation graphic and the page associated with it.

Summary

The summary page reports trading activity for the period you specify, defaulting to activity within the last hour. It also shows the default delivery exchange for receiving messages from partners and certificate information.

Properties

The properties page displays the name of the community. You can change the name. But we recommend changing a community name only after due consideration to possible effects on trading partners.

Certificates

The certificates page shows certificates associated with a community. You also can add a certificate. For more information see Manage certificates on page 325.

Routing IDs

The routing IDs page shows the routing IDs associated with a community. A community routing ID is the “from” address used in message trading. A community must have at least one, but can have multiple routing IDs. For more information see Routing IDs on page 150.

Contact

The contact page identifies the contact person for the community and the contact’s e-mail address. Other information optionally can be added, including a phone number and notes.

You can use any e-mail address you want. This can be an address for one person or an alias address with a distribution to many people. You also can enter multiple e-mail addresses by separating each address with a semicolon.

Integration delivery

The integration delivery page shows the transports set up to route messages from partners to a back-end system. A community can have multiple integration delivery exchanges. For more information see Integration exchanges on page 185.

Integration pickup

The integration pickup page shows the transports set up to retrieve messages from back-end systems for packaging and sending to partners. If a community has multiple integration pickup exchanges, all are polled for outbound messages. For more information see Integration exchanges on page 185.

Message handler

The message handler page lets you set up message actions, such as re-routing, triggered by specified conditions. For more information see Message handling on page 405.

Message validation

The message validation page lets you set whether a community accepts or rejects EDI messages with duplicate control IDs. By default duplicate messages are rejected. For more information see Inbound message validation on page 399.

Collaboration settings

The collaboration settings page lets you set up encryption, signing and other rules for messages a community sends. Provided a community uses a certificate, the default settings are adequate in many cases. For more information see Collaboration settings on page 367.

Pickup/Delivery exchange

The pickup/delivery exchange page shows the message protocols and transports partners use to send messages to the community and how the community retrieves the sent messages. A community with multiple delivery exchanges gives partners multiple avenues for sending messages. For more information see Trading delivery exchanges on page 183.

HTTP proxy

The HTTP proxy page lets you define a global HTTP proxy through which all outbound HTTP traffic is routed, if needed. All communities use this proxy. For more information see Outbound HTTP proxy on page 305.

Trading partners

The trading partners page shows the partners that belong to a community.

Add a community

There are several ways to add a community. The profile wizard prompts you for the required information.

Manually create a community profile

If you choose this option, the system prompts you to provide much of the required configuration. Other configuration is set up by default. For a list of all components that make up a community profile, see Anatomy of a community profile on page 143

Import profile information from a file

This method lets you import a profile that was previously exported as a community profile. This method is for importing a profile one at a time. For more information see Import company profile as community profile on page 152.

Check list for community configuration

The following are items to consider when configuring a community.

1. Does your community have at least one routing ID? This is the unique identifier for a community in e-commerce trading. See Routing IDs on page 150.

   If you plan to trade ebXML documents, enter an ebXML party ID type only if the routing ID is not a URI. See Routing ID for ebXML on page 460.

2. What kind of certificate do you want to use, self-signed or CA? See Manage certificates on page 325.

3. What integration delivery exchange do you want to use for picking up messages from a back-end system? See Integration exchanges on page 185.

4. What trading delivery exchange to you want to use for receiving messages from partners? See Trading delivery exchanges on page 183.

5. What integration delivery exchange to you want to use for routing messages received from partners to a back-end system? See Integration exchanges on page 185.

6. Have you added a trading partner, either by importing a profile file or manually configuring a profile? See Add a partner on page 150.

7. Have you determined whether messages sent or received by you or partners must pass through proxy servers, firewalls or load balancers? If so, take steps to adjust the trading engine configuration to accommodate unhindered message traffic. See Firewalls and proxy servers on page 155 and, if applicable, Outbound HTTP proxy on page 305

8. Do you want the community to use default or custom collaboration settings? These are rules for how outbound messages are packaged. A packaged message is one that has been signed and encrypted and also MIME-wrapped with sender and receiver information along with other data. See Collaboration settings on page 367.

9. Have you viewed the collaboration settings between your community and your partner to make sure security and packaging are aligned? See View settings between partners on page 368.

10. What document packaging rules do you want to enforce for the messages your community receives from partners? If trading EDI documents, do you want the community to accept or reject duplicate documents? See Inbound message validation on page 399.

11. Do you want to set up a post-processing script for special handling of documents before sending to partners or after receiving from partners? For example, inbound documents can be channeled to a specific integration directory with a post-processing script. See Post-processing configuration details on page 222.

12. Do you want to set up conditions for copying or rejecting messages? See Message handling on page 405.

13. Have you configured the global external SMTP server for sending e-mail messages? See Configuring external SMTP server on page 24.

14. How long do you want to retain records of processed messages? Delivery exchanges by default are set to back up all messages passing through them. Messages must be backed up to use Message tracker properly. You can set a schedule for how long records are retained. See Data backups and deletes on page 421.

The partner profile

Just as you send a community profile to your partner, your partner exports a community profile to a file and sends it to you. You import this file as the partner's profile on your system. The imported profile consists of contact information and the message protocols and transports your partner supports for receiving documents.

For partners who do not use Cyclone Commerce software, you must manually create partner profiles on your system.

Partner profiles can be present in the system without being associated with a community, but partners must belong to a community for trading to occur.

Significant elements of a partner profile are:

- The profile name.
- A routing ID to use when sending documents to the partner. A partner can have multiple routing IDs.
- One or more delivery exchanges for sending documents to the partner over the Internet.
- For secure trading, the partner's certificate and public key for encrypting the messages you send.

Anatomy of a partner profile

The user interface displays a labeled graphic that represents the parts of a partner profile. The graphic (Figure 20) appears at the top of the partner summary page and other partner-related pages. Place your cursor over a label and a red box appears. Click inside the red box to go to the page the label references.

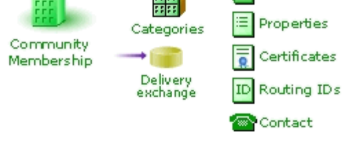


Figure 20. Partner navigation graphic

The following defines the role of each labeled part of the navigation graphic and the page associated with it.

Community membership

The community membership page lets you select the communities a partner belongs to. A partner should belong to at least one community.

Categories

The categories page lets you organize partners into groups. Use of this feature is optional. For more information see Partner categories on page 166.

Delivery exchange

The delivery exchange page shows the message protocols and transports a community uses to send messages to a partner. A partner can have multiple delivery exchanges, but only the first in the list is used. If you intend to trade with a single partner using multiple message protocols (for example, AS1 and AS3), set up one partner profile per message protocol. For more information see Trading delivery exchanges on page 183.

Summary

The summary page shows the default delivery exchange for sending messages to partners and certificate information.

Properties

The properties page displays the name of the partner. You can change the name. We recommend changing a partner name only after considering possible effects on trading.

Certificates

The certificates page shows certificates associated with a partner. You also can add a certificate. For more information see Manage certificates on page 325.

Routing IDs

The routing IDs page shows the routing IDs associated with a partner. A partner routing ID is the “to” address used in message trading. A partner must have at least one, but can have multiple routing IDs. For more information see Routing IDs on page 150.

Contact

The contact page identifies the contact person for the partner. Other information optionally can be added, including a phone number, e-mail address and notes.

Add a partner

There are several ways to add a partner. The profile wizard prompts you for the required information.

Manually create a partner profile

If you choose this option, the system prompts you to provide much of the required configuration. Other configuration is set up by default. For a list of all components that make up a partner profile, see Anatomy of a partner profile on page 148

Import profile information from a file

This method lets you import a profile that was previously exported as a partner profile. This method is for importing a profile one at a time. For more information see Import a partner profile on page 152.

Routing IDs

The trading engine lets you specify virtually an unlimited number of routing IDs for communities and partners. A routing ID is a unique identifier the trading engine uses as the “to” and “from” address for e-commerce messages exchange over the Internet.

When you add a community or partner profile, the system prompts you to enter one routing ID. After the profile is set up, you can add as many as you want. To add a routing ID, click **Routing IDs** on the navigation graphic at the top of the community or partner summary page and follow the prompts.

A routing ID can be in any format or length (up to 255 characters), including standard EDI or custom formats that include special characters or spaces.

If you use the ebXML message protocol, see Routing ID for ebXML on page 460 for special requirements.

Exporting and importing profiles

For ease of exchanging profile data, you can export a community profile and public key certificate to an XML file and give it to partners who use Cyclone Interchange or Activator 4.2.x or later. For partners who use other interoperable trading software, consult with them on the data they require of you to establish trading relationships.

Likewise, a partner who uses Cyclone Interchange or Activator 4.2.x or later can export a community or company profile and public key certificate to a file and give it to you to import as a partner profile. For partners who use other interoperable software, collect the trading data you need and then create a profile for the partner in the user interface. Partner data form on page 162 provides a way for documenting the data needed to create a partner profile.

If you are upgrading from Cyclone Interchange or Activator 4.2.x, you can export company profiles as such and import them as community profiles in this version of the trading engine.

The following topics explain how to export and import profiles. These topics address how to export or import profiles one at a time, which is a typical way of handling profiles. The topics are:

- Export a community profile
- Import a partner profile on page 152
- Import company profile as community profile on page 152

There also is a way to have the trading engine import profiles on its own without user intervention. That method is explained in Automatic profile imports on page 153.

The trading engine exports a community profile as a partner profile XML file. A community profile cannot be exported as a community profile. There also is no option in the user interface for exporting a partner profile as a partner profile, but there is a command-line tool for this (see exportProfile on page 46).

Export a community profile

Before exporting a community profile to a file as a partner profile, make sure the profile has been completely configured. On the community summary page, click **Export this community’s profile** at the bottom of the page. Save the file to a directory of your choosing. If you have associated a certificate with the profile, the certificate and public key exports with the profile.

Distribute the profile to partners by a secure means. If you send the profile file to partners as an e-mail attachment, we recommend compressing it with WinZip or similar software to ensure file integrity.

You only can export a community profile in a form usable as a partner profile. You cannot export the profile as a usable community profile.

If you give the profile file to a partner who uses Cyclone Interchange or Activator 4.2.x or earlier, the partner's system will prompt for additional information in the imported partner profile. The partner's system will prompt for the community's address, city and state or zip code. This information is not part of a community profile. You can either provide the partner with this information or the partner can determine what to enter upon importing the profile.

Import a partner profile

Upon receiving a profile from a partner, make sure the profile file is accessible on your file system. Go to the partners area of the user interface and click **Add a partner**. Then click **Import the profile information from a file**. Point the browser to the file to import, select a community for the partner and click **Finish**.

If you are expecting the imported profile to include the partner's certificate and public key, check whether a partner's certificate is trusted, go to the partner summary page and click **Certificates** in the navigation graphic at the top of the page. Click the certificate name and then click the **Trusts** tab. Check the details of third-party certificates imported with profiles to make sure trusted roots are present.

After importing a profile, go to the partner summary page for the imported partner and check whether any tasks are required to complete the profile configuration. See Post-import tasks on page 39.

If a partner uses a trading engine other than Cyclone Interchange or Activator 4.2.x. or later, you cannot import a usable partner profile, but must create the profile on the system. Partner data form on page 162 provides a way for documenting the data needed to create a profile.

Import company profile as community profile

You can export company profiles from Cyclone Interchange or Activator 4.2.x and import them as community profiles in this version of the trading engine.

First, export the company profile from Cyclone Interchange or Activator 4.2.x. You must export the profile as an XML company profile, not a partner profile. When exporting the profile, you should apply a password to protect the personal certificate.

Then go to the trading configuration area of the user interface of this version of the trading engine and click **Add a community**. Select **Import the profile information from a file**, which imports from a single file on your file system, and click **Finish**.

After importing a profile, go to the community summary page for the imported community and check whether any tasks are required to complete the profile configuration. See Post-import tasks on page 39.

Automatic profile imports

The trading engine on its own will import community or partner profiles written to a certain system directory. These must be profiles that have been exported from Cyclone Interchange or Activator 4.2.x or later or are compatible (see Creating profiles outside the application on page 154).

There are two use cases for automatic profile imports. The first is to migrate community and partner profiles from an earlier to a later version of the trading engine. This is covered in Migrating version 4.2 trading profiles on page 37.

The second case is to help manage profiles by providing a hands-free method to grow the number of partners in a community or increase the number of communities or both.

Profiles written to [install directory]\[build number]\profiles\autoImport are imported by the trading engine when the server is running. Once imported, the system moves the profiles files from profiles\autoImport to the appropriate profiles\backup subdirectory, either community or partner. The system makes all imported partner profiles members of all communities. The system creates pickup and delivery integration exchanges for an imported community.

If the trading engine is unable to import a profile file in the autoImport directory, the system moves the file to \profiles\autoImport\rejected.

The system also has some profile staging directories, as illustrated in Figure 21. The system does not write to these directories, except during installation. The directories are a place to hold profile files before a user moves them to the autoImport directory. The software installer, for instance, writes profiles files to the staged directories if the user during installation chooses to have profiles exported from an earlier version of the trading engine.

\profiles	**\autoImport**	**\rejected**
	\backup	**\community** **\partner**
	\staged	**\community** **\partner**

Figure 21. Profile autoImport and related directories

When exporting a company profile from Cyclone Interchange or Activator 4.2.x for import as a community profile in the trading engine, do not apply a password when exporting the profile as a company profile. The trading engine cannot import a password-protected profile through the autoImport directory. The password protects the certificate private key. Make sure to securely handle a company profile exported without a password.

Events for profile imports and rejects are written to control node events log (hostname_cn_events.log) in the logs directory. The events are:

Administration.Configuration.Party.CommunityImported

Administration.Configuration.Party.PartnerImported

Administration.Configuration.Party.ImportFailed

Creating profiles outside the application

In addition to using the trading engine to create, export and import profiles, you can generate profiles outside of the application by using the profile schemas. For example, you could generate profiles using your own partner management system and import them to the trading engine.

The following are the schemas to use for creating profiles:

http://www.cyclonecommerce.com/Schemas/2001/09/InterchangePartnerConfig.xsd

http://www.cyclonecommerce.com/Schemas/2001/09/InterchangeCompanyConfig.xsd

http://www.cyclonecommerce.com/Schemas/2001/08/CycloneOrganizationProfile.xsd

Firewalls and proxy servers

Many organizations have firewalls to prevent unauthorized access to their computer networks. A firewall is a server placed outside of a network. The firewall intercepts all inbound connections from the Internet, allowing only authorized users to connect to a server on the organization's network. In addition, a firewall may limit the outbound connections you can make.

It is likely you or your partners have firewalls to guard against unauthorized connections. You must take firewalls into account when configuring the trading engine.

Moreover, your network may require outbound HTTP messages to the Internet to go through a proxy server on your network. On rare occasions, the messages you send may have to go through a proxy server on your partner's network.

CAUTION: Message trading can fail if firewalls or proxy servers are not considered when configuring the trading engine. This is a common issue for new users. Consult with your network administrator if you need help with firewalls or proxy servers.

Figure 22 shows where a firewall or proxy server could be located in proximity to your or your partner's network.

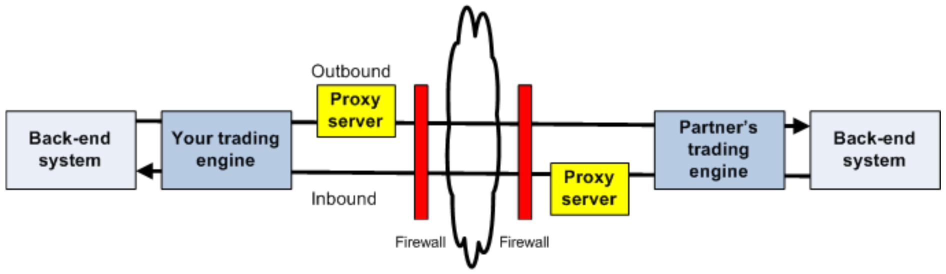


Figure 22. Possible locations for firewalls or proxy servers

As a general rule, your firewall must be configured to allow outbound HTTP traffic on the port you specify in the URL for your partner (for example, 4080). Your partner's firewall must be configured to allow inbound HTTP traffic on the port you specify.

In highly secure environments you may wish to set up firewall rules that only allow outbound HTTP traffic on this port to the IP address of your partner. However, this imposes a firewall maintenance burden on you if your partner's IP address changes or if you add partners. The same applies to your partners if they choose to configure their firewalls to allow inbound traffic only from specific IP addresses.

As part of normal operation, the operating system's socket layer dynamically allocates a local port for each outbound connection you make. This requirement is a fundamental part of socket-based protocols such as Telnet, FTP, and HTTP. It is not specific to Cyclone Activator. For example, if an outbound connection is made to an FTP host on port 21, the operating system allocates a dynamic port for the client's end of the connection. This can be seen by running the **netstat -an** command on the client after the outbound connection is established. If your firewall is so strict that it checks the ports in each packet that passes through it, you must configure the firewall to allow packets containing dynamic ports associated with local addresses. These are typically in the range of 49152 to 65535 with most operating systems, but on some systems the range is 1024 to 65535. These dynamic ports are associated only with outbound connections. It is not necessary to allow new inbound connections on these ports.

Cyclone Activator in a firewall environment

Figure 23 depicts a standard architecture for deploying Cyclone Activator in an environment where firewalls are present. To maintain document and back-end security throughout the entire process, we recommend placing the transport servers in a demilitarized zone (DMZ) and Cyclone Activator in the data layer. A DMZ is the area between an organization’s trusted internal network and an untrusted, external area such as the Internet.

If you place Cyclone Activator in the DMZ, take precautions to move the decrypted documents out of the DMZ to a secure location. You can accomplish this any number of ways. The method usually depends on your back-end needs and choice of integration options.

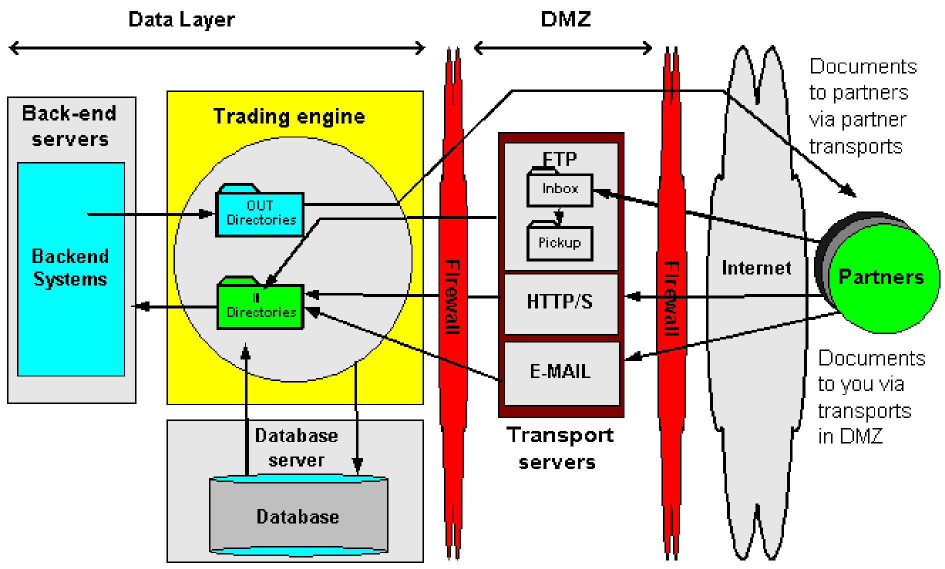


Figure 23. Standard firewall architecture

Editing URLs to allow for firewalls

If you use the embedded HTTP or HTTPS inbound transport in your community profile, you may have to make sure your partners have the right URL. This is because the URL the trading engine uses may not be the one your partners need to send documents to you through your company's firewall or load balancer or both.

When you configure the embedded HTTP or HTTPS inbound transport, the default local URL is in the following format:

```
http://<cluster machines>:4080/exchange/[routing ID]

https://<cluster machines>:4080/exchange/[routing ID]
```

The local URL contains the internal name (cluster machines) for the computer running the trading engine. You cannot change the local URL. If you installed the trading engine behind a firewall, you must make sure your partners have the external fully qualified domain name or IP address to send you documents. Depending on your transport, your partner needs a URL in the following format:

```
http://[name or IP address]:4080/exchange/[routing ID]

https://[name or IP address]:4080/exchange/[routing ID]
```

You may have to contact your company's firewall administrator to obtain this external name or IP address. This is the global IP address alias of the computer running the trading engine that is exposed to the public side of the Internet.

You can change the URL for partners on the transport maintenance page of the user interface. For more information see HTTP maintenance on page 255. After confirming the URL is correct, you can export your community profile for your partner to import as a partner profile. The external URL is contained in the partner profile.

Getting a partner's firewall information

If you will send documents via HTTP or HTTPS, contact your partner to determine whether your community needs to send documents through a firewall on the partner's network to reach the partner's trading engine. If your partner uses Cyclone Activator 5 or later, the partner may already have provided the correct public URL in the profile the partner sent you to import as a partner profile on your trading engine.

Ask the partner for the name or IP address and port number of the firewall for each transport protocol you intend to use. Also, ask whether the partner's firewall requires user name and password authentication.

In the partner profile, open the maintenance page for the transport for sending messages to the partner. Make sure the URL for sending is correct on the settings tab. Enter a user name and password to connect if required. See HTTP maintenance on page 255.

Proxy servers

If your network requires all outbound HTTP traffic to navigate a proxy server to access the Internet, you can enable this in the community profile. For more information see Outbound HTTP proxy on page 305.

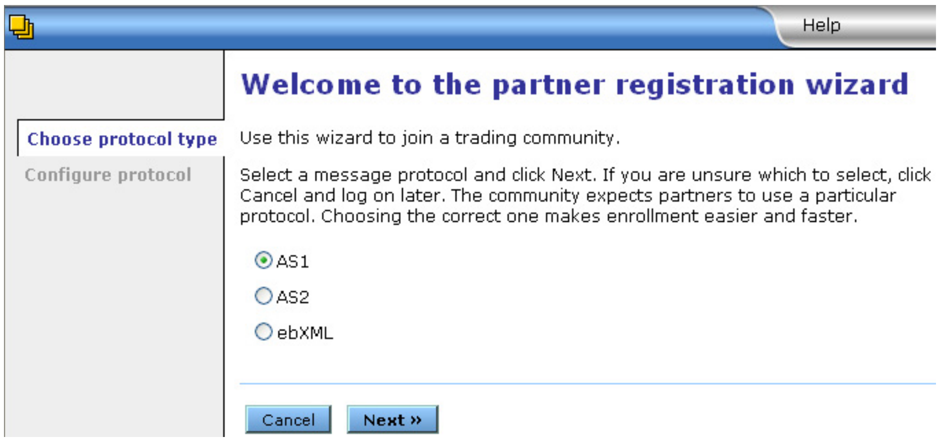
In the event your partner requires the messages you send to navigate a proxy server, see Proxy tab on page 290 for configuration information. Most users do not need to define a partner proxy, as an inbound proxy, if present, usually is transparent to the sender. In other words, proxies on the inbound side usually are for conditions such as reverse proxies for load balancing. Check with your partner to find out whether a partner's inbound proxy requires configuration on your end.

Self-registration of AS1, AS2 partners

The registration wizard helps partners of a trading community generate partner profiles for use by a Cyclone Interchange partner. This topic is for partners who want to trade via the AS1 or AS2 message protocol. For ebXML, see Self-registration of ebXML partners on page 483.

A likely use of the registration wizard is for partners who have a trading engine other than Cyclone Interchange or Activator. The wizard prompts a user to supply the information Cyclone Interchange needs to build a valid partner profile. A community, however, could have partners who use Cyclone Interchange or Activator enroll through the wizard, as well as partners who use some other interoperable trading engine.

The registration wizard is a web site hosted on the trading engine server. A community with license permissions to use the wizard gives the partner a URL to access the web site in a browser and a user name and password to log on.



Welcome to the partner registration wizard

Use this wizard to join a trading community.

Select a message protocol and click Next. If you are unsure which to select, click Cancel and log on later. The community expects partners to use a particular protocol. Choosing the correct one makes enrollment easier and faster.

[✓ AS1] AS2 ebXML

Figure 24. First page of the partner registration wizard

The following topics describe how a Cyclone Activator community prepares the partner registration wizard for use and the partner's steps for using the wizard.

Wizard preparation

Aside from configuring a community profile in the usual way, complete the following tasks so partners can join your community using the partner wizard. These steps are applicable only if your user license allows you to use the partner registration wizard.

1. Set a password for the **partner** user, if this has not already been done.

   When you log on to the user interface for the first time after installing, there is a link on the getting started page for **Set a password for partner self-registration**. Click the link and type a password for the **partner** user. This link only appears if your user license allows you to run the partner registration wizard.

   The system creates the partner user for you. Later, your partners will log on to your server's registration wizard with the user ID **partner** and the password you specify.

   If the partner user already has been set up, check the users and roles area. Select **Users and roles > Manage users** or **Users and roles > partner registrant**.

2. Give your partners the following information:

   URL

   The URL for connecting to the page for logging on to the registration wizard. The URL is in the following format:

   http://host:6080/ui/

   The variable **host** is the fully qualified domain name or IP address of the computer running the trading engine.

   User name and password

   The user name and password the partner must use for logging on to the registration wizard. Have the partner use **partner** and the password you specified for the partner user.

   Community name

   The name of the community the partner should select to join in the registration wizard.

3. Discuss with your partner whether you will trade messages via the AS1 or AS2 message protocol.

4. After a partner registers via the wizard, a message displays on the user interface home page, prompting you to approve the registration and associate the partner with your community. Click **Trading Partners** in the navigation graphic at the top of the community summary page, click **Add a partner to this community**, select **Choose an existing partner profile** and click **Next**. Select the partner and click **Add**.

Using the partner wizard

Before using the partner wizard, you need the following information at hand:

1. The URL to connect to the wizard in a browser.
2. A user name and password to log on to the wizard.
3. The name of the community you will join.
4. The name of your company.
5. A company contact name and e-mail address.
6. The routing ID to use as your company's unique identifier for e-commerce trading.
7. An encryption certificate and public key exported to a file with an extension of .cer, .p7b or .p7c. Do not use a certificate file that contains your private key.
8. If trading via AS1, the e-mail address you want the community to use for sending messages to your company.
9. If trading via AS2, the HTTP or HTTPS URL the community needs for sending messages to your company. If HTTPS, you have the option of specifying in the wizard whether to compare the name of the SSL server to the name in the server's certificate to ensure they are the same. Also, if the community needs a user name or password to connect to the server, you must provide that information in the wizard.

Once you have collected this information, do the following.

Steps

1. Log on to the partner registration wizard.
2. Follow the wizard prompts for registering.
3. Wait for the community to contact you with the next steps.

Partner data form

Use the following form for recording data about a partner who uses a trading engine other than Cyclone Interchange or Activator 5.0 or later or Cyclone Interchange or Activator 4.2.x. You can use the information on the completed form to create a partner profile for the partner in the user interface.

When you are ready to add a partner profile, go to the partners area of the user interface, click **Add a partner** and then click **Manually create a new partner profile**.

Trading engine

Field	Partner’s data
Product name	
Version number	

Identity

Fields followed by an asterisk represent required information in Cyclone Activator.

Field	Partner’s data
Company name *	
Routing ID * One ID is required but the partner can have multiple IDs	
Contact name *	
Contact phone number *	
Contact e-mail address *	

Protocol

The partner’s preferred protocol for your community to send documents.

Protocol	Partner's data
EDIINT AS1 (e-mail)	
EDIINT AS2 (HTTP)	
Secure file (HTTP, FTP, file system, JMS, MQSeries). Files are packaged using S/MIME.	
Secure e-mail (for partners who use mail clients such as Microsoft Outlook)	
Other (FTP, file system, JMS, MQSeries). No packaging or security	
ebXML	

Transport details

The partner's preferred transport for your community to send documents.

HTTP, e-mail

Field		Partner's data
HTTP	URL	
HTTPS	URL	
	Authenticate SSL connection	Circle one: yes no
SMTP/POP	e-mail address	
SMTP-to-SMTP	e-mail address	
	SMTP server name	
	Port (default 25)	
	Use SSL	Circle one: yes no
	If use SSL is yes, the SSL port	

FTP

Setting	Partner's data
FTP server name	

Setting	Partner's data
Port	
Inbox	
Pickup	
User name	
Password	
Clients must use SSL to connect to this server	

JMS

Setting	Partner's data
JNDI URL	
JNDI factory	
JNDI user name	
JNDI password	
JMS queue	
JMS connection factory	
JMS user name	
JMS password	

MQSeries

Setting	Partner's data
MQSeries server	
Port (default 1414)	
Queue name	
Queue manager	
Channel	
Character set (default 819)	
User name	
Password	

Preferences

Preference	Partner's data
Preserve original filenames	
If preserve original filenames is true, overwrite duplicate filenames	
If preserve original filenames is true, sequentially number duplicate filenames	
Generate unique filenames	

Security details

The partner must provide a digital certificate to effect secure trading.

Field	Partner's data
Sign documents	Circle one: yes no
Request acknowledgments	Circle one: yes no
Request signed acknowledgments	Circle one: yes no
If HTTP or HTTPS, request synchronous acknowledgments	Circle one: yes no
Message digest	Circle one: SHA1 (default) MD5
Encrypt documents (yes or no)	Circle one: yes no

Binary and XML documents

Field	Partner's data
Partner will trade binary documents	Circle one: yes no
Partner will trade XML documents	Circle one: yes no
XML type	
Sender XPath if custom XML type	

Field	Partner’s data
Receiver XPath if custom XML type	

Partner categories

The user interface provides the ability to sort partner profiles into categories. Assigning partners to categories can help you locate a specific partner more quickly on search pages when you have several partners from which to choose.

To create a partner category, select **Manage categories** from the partner menu, and then select **Add a category**. Type a category name, and optionally select a parent category.

To add a partner to a category, click **Categories** from the partner’s summary page, and then select the check box for the category to add the partner to.

❖ ❖ ❖

12 Embedded transport servers

The trading engine lets communities use embedded HTTP, SMTP and web services API transport servers with delivery exchanges. These are provided so external servers are not needed for trading.

There are two types of embedded transport servers: global and community.

The trading engine creates the global embedded transport servers. These are global because multiple delivery exchanges and communities can use them.

Community-level embedded transport servers are created when a community chooses to do so when adding an embedded HTTP or SMTP delivery exchange with the delivery exchange wizard (see The delivery exchange wizard on page 189). A community can re-use the embedded servers it has created in multiple delivery exchanges, but other communities must create their own community-level embedded servers.

Details about configuring transports are in Delivery exchanges on page 181.

Concepts

- Global embedded HTTP server on page 168
- Global embedded SMTP server on page 169
- Global embedded web services API server on page 170
- Embedded HTTP use cases on page 175

Procedure

- Change community embedded server on page 171

Pages and fields

- Community embedded HTTP fields on page 171
- Community embedded SMTP fields on page 173

Global embedded HTTP server

The global embedded HTTP transport server is available upon installation. Communities can share it. You can change the server's port and advanced options. To change settings, go to the trading configuration area of the user interface and click **Configure the global embedded HTTP server**.

This server only is for HTTP. For HTTPS use a community-level embedded server (see Community embedded HTTP fields on page 171).

The following are the configuration fields for the server.

Settings tab

Host

The fully qualified domain name of the computer on which the embedded server runs. The trading engine detects this setting; you cannot change it.

Port

The port on which the trading engine listens for connection requests.

Advanced tab

Minimum threads

The least number of threads the trading engine must dedicate to the server.

Maximum threads

The most threads the trading engine can dedicate to the server.

Maximum linger time (seconds)

The amount of time the connection is kept open after it is closed. A negative number indicates that the connection should not be kept open after it is closed.

Read timeout (seconds)

How many seconds of inactivity to allow before the trading engine terminates the connection.

Global embedded SMTP server

The global embedded SMTP transport server is available upon installation. Communities can share it. You can change the server's port and advanced options. To change settings, go to the trading configuration area of the user interface and click **Configure the global embedded SMTP server**.

The following are the configuration fields for the server.

Settings tab

Host

The fully qualified domain name of the computer on which the embedded server runs. The trading engine detects this setting; you cannot change it.

Port

The port on which the trading engine listens for connection requests.

Host name used by partners

The fully qualified domain name or IP address that a community's partners must use to connect to this embedded server. The trading engine supplies a value based on the name of the host computer. It's possible you may have to change it. Contact your firewall administrator if you need help with this field.

Port used by partners

The port number that a community's partners must use to connect to this embedded server. Contact your firewall administrator if you need help with this field.

Advanced tab

Backlog

The number of connections that the server puts "on hold" while it is busy. Once this number is reached, connections are refused.

Read timeout (seconds)

How many seconds of inactivity to allow before the trading engine terminates the connection.

Global embedded web services API server

The global embedded web services API server is available upon installation. Communities can share it. You can change the server's port and advanced options. To change settings, go to the trading configuration area of the user interface and click **Configure the global embedded web services API server**.

For information about how to configure the server for use in trading, see Web services API transport on page 221.

The following are the configuration fields for the server.

Settings tab

Host

The fully qualified domain name of the computer on which the embedded server runs. The trading engine detects this setting; you cannot change it.

Port

The port on which the trading engine listens for connection requests.

Advanced tab

Minimum threads

The least number of threads the trading engine must dedicate to the server.

Maximum threads

The most threads the trading engine can dedicate to the server.

Maximum linger time (seconds)

The amount of time the connection is kept open after it is closed. A negative number indicates that the connection should not be kept open after it is closed.

Read timeout (seconds)

How many seconds of inactivity to allow before the trading engine terminates the connection.

Change community embedded server

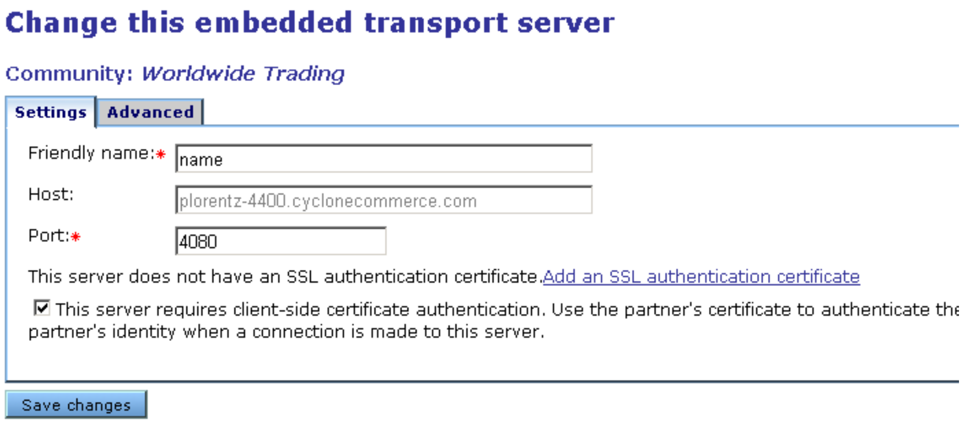
After setting up a delivery exchange with a community-level embedded SMTP or HTTP server, you can change its settings. The embedded transport maintenance page is accessible from a community's summary page. Because embedded transport servers can be shared between different delivery exchanges, changes you make to the servers are reflected in all delivery exchanges that use them.

Steps

1. From the community summary page, click **Change an embedded transport server**. A list of embedded transports displays.
2. Click the embedded transport server you want to change.
3. Edit the values on the Settings or Advanced tabs as necessary, and then click **Save changes**. For field descriptions see Community embedded HTTP fields on page 171 or Community embedded SMTP fields on page 173.

Community embedded HTTP fields

The following are the maintenance fields for a community embedded HTTP or HTTPS transport server.



Community: *Worldwide Trading*

Friendly name:* name

Host: plorentz-4400.cyclonecommerce.com

Port:* 4080

This server does not have an SSL authentication certificate.Add an SSL authentication certificate

☑ This server requires client-side certificate authentication. Use the partner's certificate to authenticate the partner's identity when a connection is made to this server.

Figure 25. Community embedded HTTPS server settings tab

Settings tab

Friendly name

A name you give the transport server to distinguish it from other community-level embedded servers. This field gets its initial value when you type it in the delivery exchange wizard.

Host

The fully qualified domain name of the computer on which the embedded server runs. The trading engine detects this setting; you cannot change it.

Port

The port on which the trading engine listens for connection requests.

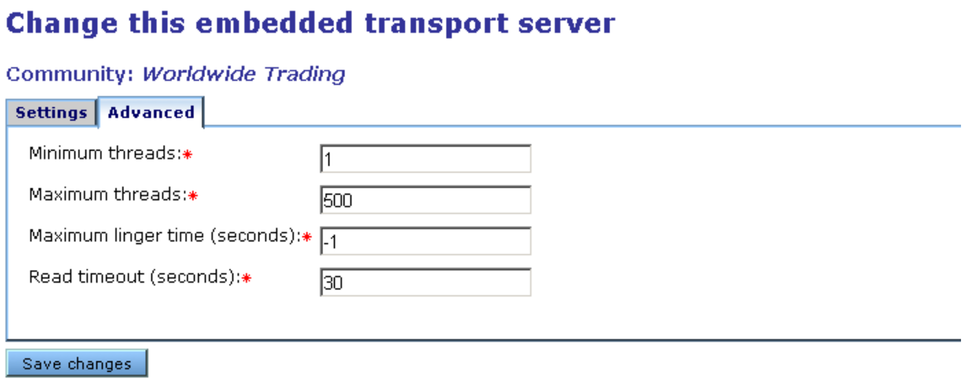
The following display only for an HTTPS server.

[SSL certificate message]

An HTTPS server requires an SSL certificate. If the server has a certificate, the name of the certificate is displayed. If the server does not have an SSL certificate, you are prompted to provide one.

This server requires client-side certificate authentication

Select this to use the partner's certificate to authenticate the partner when the partner connects to the server.


Change this embedded transport server

Community: *Worldwide Trading*

Minimum threads:*	1
Maximum threads:*	500
Maximum linger time (seconds):*	-1
Read timeout (seconds):*	30

Figure 26. Community embedded HTTP server advanced tab

Advanced tab

Minimum threads

The least number of threads the trading engine must dedicate to the server.

Maximum threads

The most threads the trading engine can dedicate to the server.

Maximum linger time (seconds)

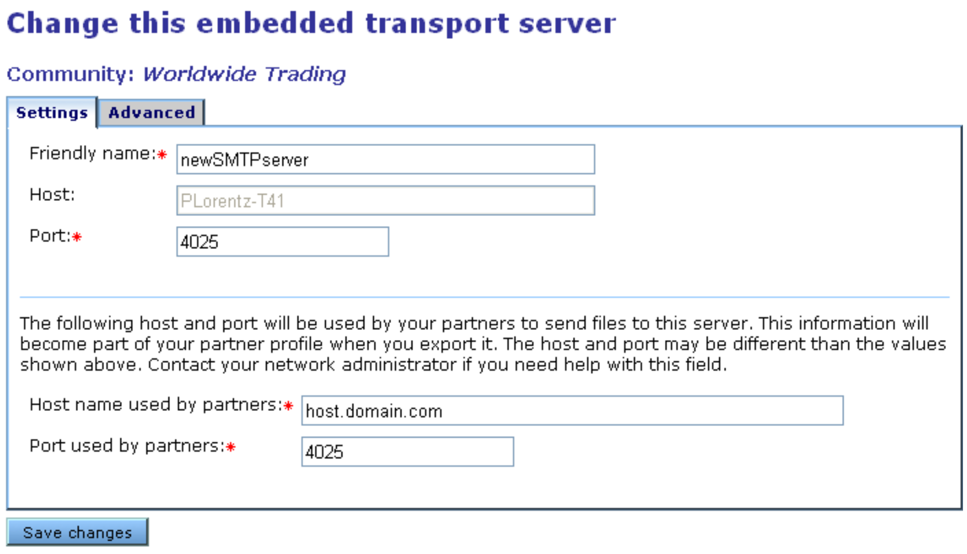
The amount of time the connection is kept open after it is closed. A negative number indicates that the connection should not be kept open after it is closed.

Read timeout (seconds)

How many seconds of inactivity to allow before the trading engine terminates the connection.

Community embedded SMTP fields

The following are the maintenance fields for a community embedded SMTP transport server.


Change this embedded transport server
Community: Worldwide Trading
Settings
Advanced
Save changes

Friendly name:*	newSMTPserver
Host:	PLorentz-T41
Port:*	4025

The following host and port will be used by your partners to send files to this server. This information will become part of your partner profile when you export it. The host and port may be different than the values shown above. Contact your network administrator if you need help with this field.

Host name used by partners:*	host.domain.com
Port used by partners:*	4025

Figure 27. Community embedded SMTP server settings tab

Settings tab

Friendly name

A name you give the transport server to distinguish it from other embedded servers. This field gets its initial value when you type it in the delivery exchange wizard.

Host

The fully qualified domain name of the computer on which the embedded server runs. The trading engine detects this setting; you cannot change it.

Port

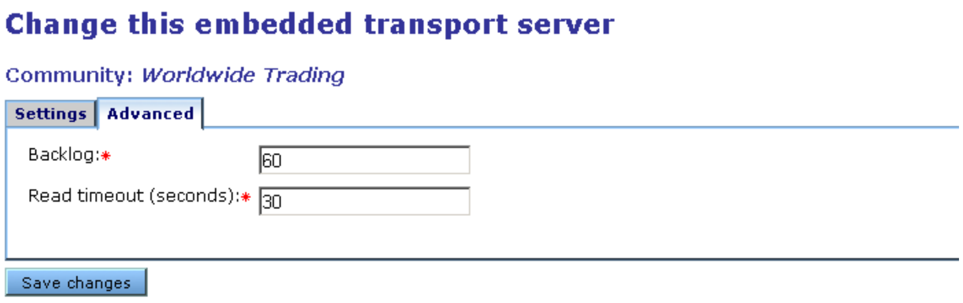
The port on which the trading engine listens for connection requests.

Host name used by partners

The fully qualified domain name or IP address that a community's partners must use to connect to this embedded server. The trading engine supplies a value based on the name of the host computer. It's possible you may have to change it. Contact your firewall administrator if you need help with this field.

Port used by partners

The port number that a community's partners must use to connect to this embedded server. Contact your firewall administrator if you need help with this field.


Change this embedded transport server

Community: *Worldwide Trading*

Backlog:* 60

Read timeout (seconds):* 30

Figure 28. Community embedded SMTP server advanced tab

Advanced tab

Backlog

The number of connections that the server puts “on hold” while it is busy. Once this number is reached, connections are refused.

Read timeout (seconds)

How many seconds of inactivity to allow before the trading engine terminates the connection.

Embedded HTTP use cases

This topic provides examples of using various remote URL and proxy settings to achieve different results with the embedded HTTP transport servers. These use cases apply to the global and community embedded HTTP transport servers.

The trading engine user interface provides flexibility for several setup variations. The use cases include the most common combinations.

Case A

Embedded HTTP

A partner connects to host:port from a remote URL and sends a POST request containing the path part of the URL.

UI settings

Field	Sample setting
Local URL	http://server1.domain.com:4080/exchange/routingID
Remote URL	(same)
Proxy	(none)

Results

Action	Value
Server binds to	<cluster_machines>:4080
Partner connects to	**server1.domain.com:4080**
Partner request	POST **/exchange/routingID** HTTP/1.1 HOST server1.domain.com:4080 <other headers and payload>

Case B

Embedded HTTP with remote URL

A partner connects to host:port from a remote URL and sends a POST request containing the path part of the remote URL. The firewall or reverse proxy maps the remote host and port to a real server. In the following example, edi.domain.com:5080 would be mapped to a server in the cluster listening to port 4080.

UI settings

Field	Sample setting
Local URL	http://server1.domain.com:4080/exchange/routingID
Remote URL	http://edi.domain.com:5080/exchange/routingID
Proxy	(none)

Results

Action	Value
Server binds to	<cluster_machines>:4080
Partner connects to	**edi.domain.com:5080**
Partner request	POST **/exchange/routingID** HTTP/1.1 HOST edi.domain.com:5080 <other headers and payload>

Case C

Embedded HTTP with remote URL including path remapping

This case is the same as the previous case, except that the remote URL has a different path than the local one, which means a proxy is present that is rewriting the path in the POST request based on an internal mapping. In the following example, the path is changed from /inbound/stuff to /exchange/routingID.

UI settings

Field	Sample setting
Local URL	http://server1.domain.com:4080/exchange/routingID
Remote URL	http://edi.domain.com:5080/inbound/stuff
Proxy	(none)

Results

Action	Value
Server binds to	<cluster_machines>:4080
Partner connects to	**edi.domain.com:5080**
Partner request	POST **/exchange/routingID** HTTP/1.1 HOST edi.domain.com:5080 <other headers and payload>

Case D

Embedded HTTP with proxy and remote URL including path remapping

This case is similar to the previous case, except the host and port of the proxy are explicitly specified. The partner connects to the proxy host:port and sends a POST request containing the full remote URL. The proxy uses this information to establish a connection to the real endpoint server. In the following example, the proxy sends the POST request for edi.domain.com:5080 to one of the servers in the cluster listening to port 4080. It also translates the external /inbound/stuff path to /exchange/routingID.

UI settings

Field	Sample setting
Local URL	http://server1.domain.com:4080/exchange/routingID
Remote URL	http://edi.domain.com:5080/inbound/stuff
Proxy	proxy.domain.com:8080

Results

Action	Value
Server binds to	<cluster_machines>:4080
Partner connects to	**proxy.domain.com:8080**
Partner request	POST **http://edi.domain.com:5080/inbound/stuff** HTTP/1.1 HOST **edi.domain.com:5080** <other headers and payload>

Case E

Embedded HTTPS (SSL) with remote URL

A partner connects to host:port from remote URL and sends POST request containing the path part of the remote URL. The firewall or reverse proxy maps the remote host and port to a real server. In the following example, edi.domain.com:1453 is mapped to a server in the cluster listening to port 1443. This example does not show path remapping, though that might be present as well.

UI settings

Field	Sample setting
Local URL	https://server1.domain.com:1443/exchange/routingID
Remote URL	https://edi.domain.com:1453/exchange/routingID
Proxy	(none)

Results

Action	Value
Server binds to	<cluster_machines>:1443
Partner connects to	edi.domain.com:1453
Partner request	POST **/exchange/routingID** HTTP/1.1 HOST **edi.domain.com:1453** <other headers and payload>

Case F

Embedded HTTPS (SSL) with proxy and remote URL including path remapping

This is similar to the non-HTTPS proxy case, except that the partner starts off by sending a CONNECT request to tell the proxy to establish a tunnel to the endpoint server. This allows it to perform SSL handshaking with the endpoint server before sending the POST request. Note in the following example that the POST request contains only the path, unlike the non-SSL proxy case, which includes the full URL in order to tell the proxy how to find the endpoint server.

HTTPS-capable clients that can handle non-transparent proxies are set up to perform these two steps (CONNECT followed by POST), which keeps the setup simpler for the user.

UI settings

Field	Sample setting
Local URL	https://server1.domain.com:1443/exchange/routingID
Remote URL	https://edi.domain.com:1453/exchange/routingID
Proxy	proxy.domain.com:8083

Results

Action	Value
Server binds to	<cluster_machines>:1443
Partner connects to	proxy.domain.com:8083
Partner request 1	CONNECT **edi.domain.com:1453**
Partner request 2	POST **/exchange/routingID** HTTP/1.1 HOST edi.domain.com:1453 <other headers and payload>

❖ ❖ ❖

13 Delivery exchanges

Delivery exchanges are the message protocols or transports or both a community or partner uses to send and receive messages. These can be set up in simple or complex configurations, depending on needs.

A message protocol is a standard or convention for handling the exchange of e-commerce business messages over the Internet, and sometimes between back-end systems and the trading engine. A message protocol supports one or more transports. With few exceptions, these are TCP/IP transports.

Concepts

Procedure

Pages and fields

Four exchanges required

You can have as many delivery exchanges, or transports, as a user license allows, but at least four are required:

1. A pickup integration exchange to retrieve messages from a back-end system. This is set up in the profile for the local community.
2. A trading exchange to send messages over the Internet to partners. This is set up in the profile for the remote partner.
3. A trading exchange to receive messages sent by partners. This is set up in the profile for the local community.
4. A delivery integration exchange to route received messages to a back-end system. This is set up in the profile for the local community.

Figure 29 illustrates this concept. The numbers in the diagram correspond to the preceding numbers.

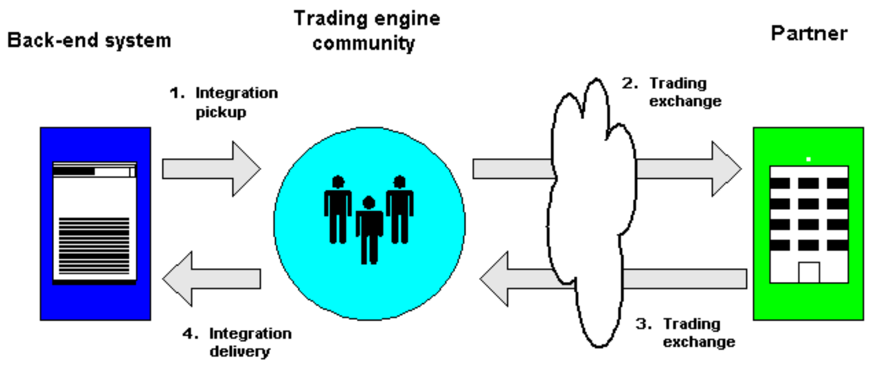


Figure 29. The four delivery exchanges

There are variants on how a community receives messages from partners. They can be described as the direct send and send-pickup methods. The direct send method is when, for example, a remote partner and a local community use the same URL to send and pick up messages from an embedded HTTP server. The send-pickup method is when a remote partner sends a message via one transport and the local community can pick up the message via another, such as SMTP and POP. The trading delivery exchange maintenance page for communities in the user interface clearly shows the protocols that use each method.

Note: This documentation often uses the term **delivery exchange** to mean all types of exchanges, including trading exchanges and pickup and delivery integration exchanges.

Trading delivery exchanges

Trading delivery exchanges in a community profile specify how you want your local community to receive documents over the Internet from remote partners. Trading delivery exchanges in partner profiles specify how a local community sends documents to remote partners.

The user interface lists the delivery exchanges used for trading by message protocol. A message protocol supports one or more transports.

Table 18 lists all of the trading delivery exchanges communities and partners can use for exchanging messages. The ones your trading engine supports depends on your user license. Communities also require pickup and delivery integration exchanges, which are explained in Integration exchanges on page 185.

Your local community and remote partner can use the same or a different message protocol for exchanging messages. However, if you use different protocols, you must be prepared to return receipts via the same protocol as the payloads were received. The partner should configure his system likewise.

Figure 30 illustrates a trading relationship where the local community and the remote partner use different message protocols. The community prefers receiving payloads via the AS1 protocol. The partner prefers receiving payloads via AS2. To accommodate these preferences, both parties must be prepared to send and receive via AS1 and AS2. In this scenario, the community receives a payload via AS1 and returns a receipt to the sending partner via AS1. Meanwhile, the community sends a payload via AS2 and receives a receipt from the receiving partner via AS2.

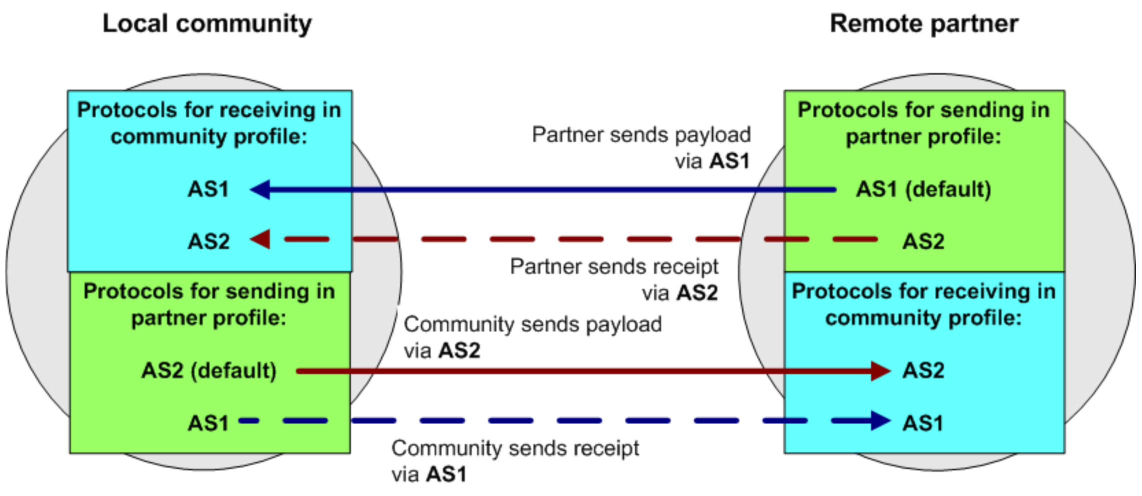

Local community
Remote partner
Protocols for receiving in community profile:
AS1
AS2
Protocols for sending in partner profile:
AS2 (default)
AS1
Partner sends payload via AS1
Partner sends receipt via AS2
Community sends payload via AS2
Community sends receipt via AS1
Protocols for sending in partner profile:
AS1 (default)
AS2
Protocols for receiving in community profile:
AS2
AS1

Figure 30. Community and partner send via different message protocols

Notice in Figure 30 that on the community side, AS2 is the designated default protocol for sending in the partner profile. Likewise, on the partner side, AS1 is the default for sending. This is necessary because the trading engine uses the default protocol for sending payloads. See View settings between partners on page 368.

The packaging options column in Table 18 indicates whether a message protocol supports signing and encryption of messages using digital certificates. It also indicates whether message compression is supported.

Table 18 - Supported trading exchanges

Message protocol	Transport options	Packaging options
EDIINT AS1	SMTP/POP Embedded SMTP	Signing Encryption Compression
EDIINT AS2	Embedded HTTP Embedded HTTPS Staged HTTP web servlet	Signing Encryption Compression
EDIINT AS3	FTP SFTP	Signing Encryption Compression
Secure file	File system FTP SFTP Embedded HTTP Embedded HTTPS JMS MQSeries	Signing Encryption Compression
Secure e-mail	SMTP/POP SMTP	Signing Encryption
Other	File system FTP SFTP JMS MQSeries	None; messages are not signed, encrypted, or compressed
ebXML	Embedded HTTP Embedded HTTPS Embedded SMTP SMTP/POP Staged HTTP web servlet	Signing Encryption
Web Services	Embedded HTTP Embedded HTTPS	

Table 18 - Supported trading exchanges

Message protocol	Transport options	Packaging options
RosettaNet 1.1	Embedded HTTP Embedded HTTPS	Signing Compression
RosettaNet 2.0	Embedded HTTP Embedded HTTPS	Signing Encryption Compression

Note: RosettaNet requires a special user license to enable the protocol in the trading engine.

References for popular message protocols

For detailed information about widely recognized protocols, go to the following sites:

AS1, AS2, AS3	http://www.ietf.org/
ebXML	www.ebxml.org or www.oasis-open.org
RosettaNet	www.rosettanet.org

Interoperability

The trading engine has been certified for interoperability for AS1, AS2, AS3 and ebXML. See www.ebusinessready.org for a list of software that the trading engine has been successfully tested with for interoperability.

Integration exchanges

Pickup and delivery integration exchanges are set up in community profiles. A pickup exchange specifies how the trading engine retrieves business documents from a back-end system for packaging and sending to partners. A delivery integration exchange specifies how the trading engine routes documents received from partners to a back-end system.

Integration exchanges do not use message protocols, as trading delivery exchanges do. When you set up integration exchanges, you simply select a transport method and messages are moved without packaging, encryption or compression.

Table 19 lists the available transports for integration pickup and delivery exchanges.

Table 19 - Supported integration exchanges

Transport	Integration pickup	Integration delivery
File system transport	✓	✓
File system with message meta-data (see File system transport)		✓
FTP transport	✓	✓
IBM MQSeries transport	✓	✓
JMS transport	✓	✓
SFTP transport	✓	✓
Web services API client (see Web services API transport)		✓
Web services API server (see Web services API transport)	✓	

A community needs at least one pickup and one delivery integration exchange, but can have multiple integration exchanges.

The file system with message meta-data transport is configured the same as File system transport, but is for when a community wants to produce MMDs to file system inbound integration with ebXML or RosettaNet documents. If a community profile is configured for the ebXML or RosettaNet message protocol and any other protocol, the default inbound integration transport cannot be file system with message meta-data. See ebXML support on page 443 or RosettaNet support on page 489.

Integration exchanges can be configured with conditions specifying senders and receivers, meta-data attributes and rules for routing inbound messages to various integration exchanges. See: From address and To address tabs on page 291, Message attributes tab on page 293 and Delivery criteria tab on page 300.

Note: This documentation often uses the term **delivery exchange** to mean all types of transports, including trading exchanges and pickup and delivery integration exchanges.

Multiple integration pickup exchanges

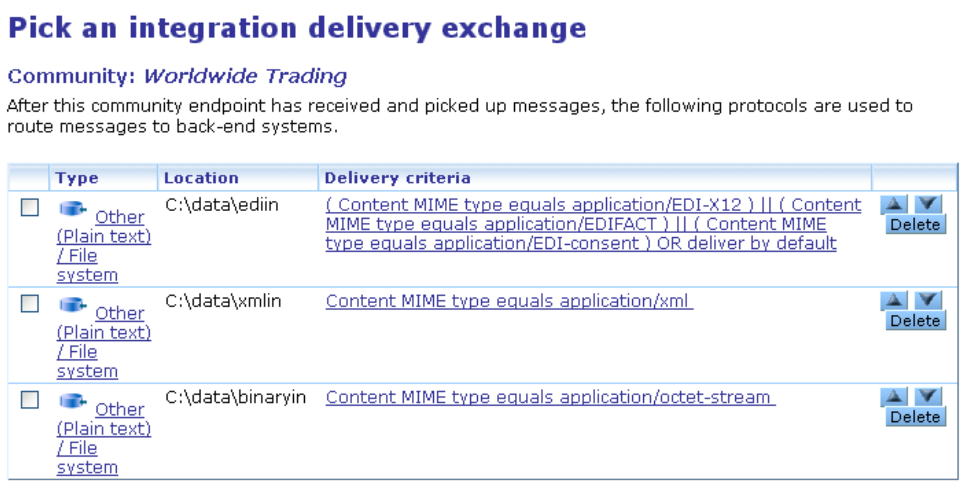
If there are multiple integration pickup exchanges, the trading engine checks all for outbound messages. What's more, all communities share all integration pickup exchanges. The trading engine determines the sender by parsing for the "from" address in the document. Alternately, you can set a fixed value for the "from" address so that all messages picked up from a particular exchange have a single community as the sender. For more information see From address and To address tabs on page 291.

Multiple integration delivery exchanges

If a community has multiple integration delivery exchanges for routing received messages to back-end systems, you can set rules that specify when to use the exchanges based on conditions. This is useful when you want to route certain messages to a particular exchange. For configuration details see Delivery criteria tab on page 300.

For example, when you set up default file system integration exchanges, the trading engine uses MIME type rules to allocate inbound messages by document type (see Set up default file system integration on page 227).

To see the integration delivery exchanges for a community, click **Integration delivery** on the navigation graphic at the top of the community summary page. The exchange at the top of the list is the default exchange. This means if an inbound message does not meet any of the conditions for other exchanges, the message is routed to this exchange. Figure 31 shows an example of a file system exchange (c:\data\ediin) with MIME rules that only EDI messages are allowed. However, the exchange, as indicated by "OR deliver by default," also is the default exchange because it is at the top of the list.



Pick an integration delivery exchange

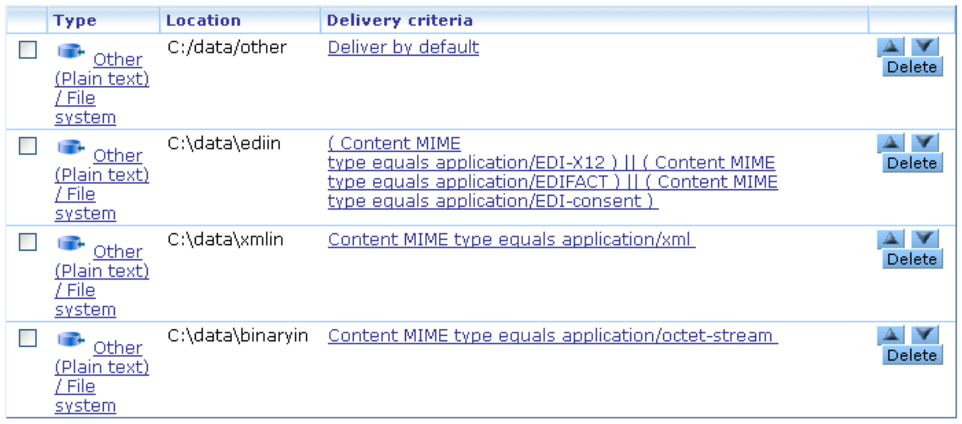
Community: *Worldwide Trading*

After this community endpoint has received and picked up messages, the following protocols are used to route messages to back-end systems.

	Type	Location	Delivery criteria	
☐	Other (Plain text) / File system	C:\data\ediin	(Content MIME type equals application/EDI-X12) || (Content MIME type equals application/EDIFACT) || (Content MIME type equals application/EDI-consent) OR deliver by default	Delete
☐	Other (Plain text) / File system	C:\data\xmlin	Content MIME type equals application/xml	Delete
☐	Other (Plain text) / File system	C:\data\binaryin	Content MIME type equals application/octet-stream	Delete

Figure 31. Exchange at top of list is the default

If a community has multiple integration delivery exchanges, an exchange without delivery criteria is not used unless it is at the top of the list. A way to make sure that inbound message that do not meet the rules for other exchanges are delivered to a single exchange is to add an exchange with no delivery criteria and position it at the top of the exchange list. Figure 32 shows that a file system exchange is the default to use when inbound messages do not meet the criteria specified by the other exchanges.



	Type	Location	Delivery criteria	
☐	Other (Plain text) / File system	C:/data/other	Deliver by default	Delete
☐	Other (Plain text) / File system	C:\data\ediin	(Content MIME type equals application/EDI-X12) || (Content MIME type equals application/EDIFACT) || (Content MIME type equals application/EDI-consent)	Delete
☐	Other (Plain text) / File system	C:\data\xmlin	Content MIME type equals application/xml	Delete
☐	Other (Plain text) / File system	C:\data\binaryin	Content MIME type equals application/octet-stream	Delete

Figure 32. Delivery exchange with no criteria at top of list is the default

Adding transports

The following is a list of all the transports the trading engine supports for moving business documents and messages such as receipts. Use this list as a quick reference to look up information about using the delivery exchange wizard to add a transport.

- Detached e-mail for community on page 191

- Embedded e-mail for community on page 192
- E-mail transport for partners on page 193
- Embedded HTTP transport on page 195
- Staged HTTP on page 235
- HTTP transport for partners on page 198
- File system transport on page 200
- FTP transport on page 203
- JMS transport on page 212
- IBM MQSeries transport on page 219
- Web services API transport on page 221

The topic for each transport documents the configuration fields displayed in the delivery exchange wizard.

After using the delivery exchange wizard to add a transport, you can use maintenance pages to change, test, activate, deactivate or delete community or partner exchanges. The wizard and maintenance pages have many of the same configuration fields. Generally, however, the wizard offers only the most commonly used fields for a transport. The maintenance pages have all configuration fields that can be changed. After using the wizard to add a transport, you might have to go the transport's maintenance page to fine tune the configuration.

Some of these transports are supported by more than one message protocol. Regardless of the affiliated message protocol, the configuration of the transport is the same. For a list of message protocols and their transports, see Trading delivery exchanges on page 183.

The delivery exchange wizard

The user interface has a wizard to help you add trading or integration exchanges to profiles. It is called the delivery exchange wizard. The wizard displays different configuration options, depending on the type of exchange you are adding.

After using the delivery exchange wizard to add a transport, you can use maintenance pages to change, test, activate, deactivate or delete community or partner exchanges. The wizard and maintenance pages have many of the same configuration fields. Generally, however, the wizard offers only the most commonly used fields for a transport. The maintenance pages have all configuration fields that can be changed. After using the wizard to add a transport, you might have to go the transport's maintenance page to fine tune the configuration.

The following topics explain how to launch the delivery exchange wizard.

Open wizard on community summary page

The following are methods to open the delivery exchange wizard on a community summary page.

- While configuring a community, the following links appear on the community summary page, prompting you to complete these delivery exchange tasks. Once the tasks are completed, the prompts disappear.

  Set up a delivery exchange for picking up messages from integration

  Set up a delivery exchange for receiving messages from partners

  Set up a delivery exchange for routing received messages to integration

- Click **Delivery exchange** on the navigation graphic at the top of the page. This opens a maintenance page listing any exchanges configured so far for the community. Click **Add a delivery exchange**.

- Click **Integration delivery** on the navigation graphic at the top of the page. This opens a maintenance page listing any exchanges configured so far for the community. Click **Add an integration delivery exchange**.

- Click **Integration pickup** on the navigation graphic at the top of the page. This opens a maintenance page listing any exchanges configured so far for the community. Click **Add an integration pickup delivery exchange**.

Open wizard on partner summary page

The following are methods to open the delivery exchange wizard on a partner summary page.

- While configuring a partner, the following link appears on the partner summary page, prompting you to complete this delivery exchange task. Once the task is completed, the prompt disappears.

  Set up a delivery exchange for sending messages to this partner

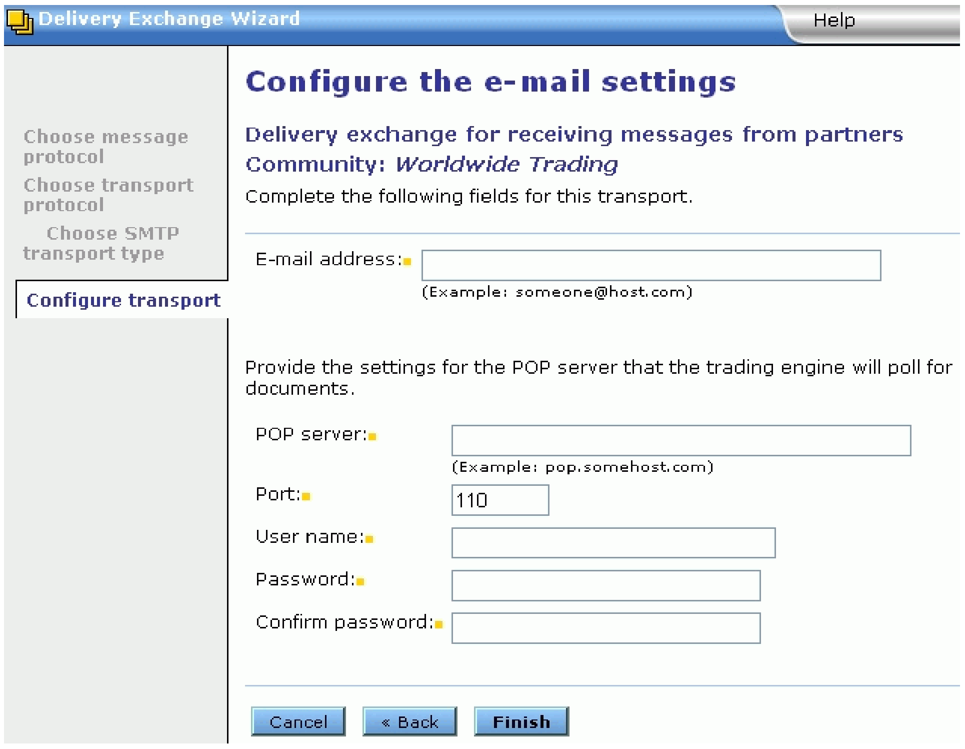
- Click **Add a delivery exchange** at the bottom of the partner summary page.

- Click **Delivery exchange** on the navigation graphic at the top of the page. This opens a maintenance page listing any exchanges configured so far for the partner. Click **Add a delivery exchange**.

Detached e-mail for community

When setting up a delivery exchange with a detached e-mail server for a community, the trading engine splits the information you provide into separate POP and SMTP settings on the maintenance page. The POP settings indicate how the trading engine retrieves e-mail messages from partners. The SMTP settings are provided to your trading partners so they know how to send messages to the community.

The term detached is used to distinguish this e-mail method, which uses an external POP server, from the system's embedded SMTP server, which is explained in Embedded e-mail for community on page 192. The delivery exchange wizard provides options for using a detached or embedded SMTP server.



Configure the e-mail settings

Delivery exchange for receiving messages from partners

Community: *Worldwide Trading*

Complete the following fields for this transport.

E-mail address:

(Example: someone@host.com)

Provide the settings for the POP server that the trading engine will poll for documents.

POP server:

(Example: pop.somehost.com)

Port: 110

User name:

Password:

Confirm password:

Figure 33. Detached e-mail server setup in delivery exchange wizard

The following are the fields in the delivery exchange wizard for configuring the detached e-mail server transport for a community.

E-mail address

The e-mail address that the remote partner uses to send messages to your local community.

POP server

The name of the POP server that the trading engine polls for messages sent by your partner. This must be a fully qualified domain name or IP address.

Port

The POP server port by default is 110.

User name

The user name to connect to the server.

Password

The password to connect to the server.

Confirm password

The password to connect to the server.

Embedded e-mail for community

A community can use an embedded SMTP server to receive messages from partners.

When you elect to set up an e-mail transport for a community using an embedded SMTP server, the delivery exchange wizard asks whether you want to:

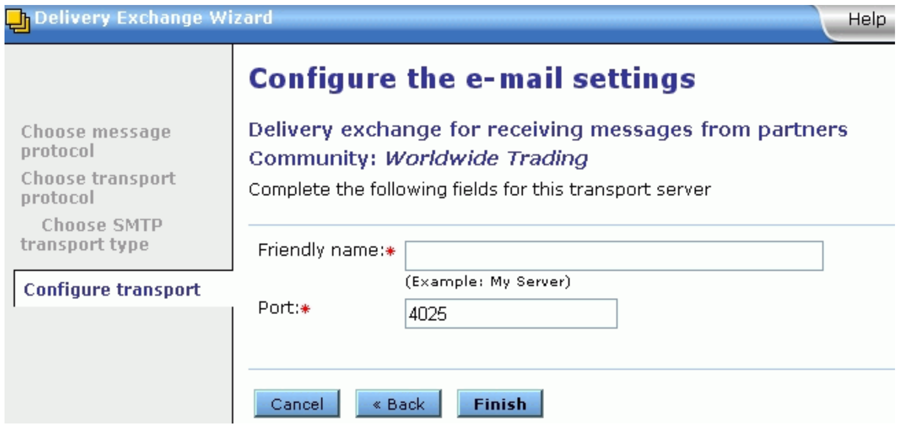
- Use the system’s global embedded SMTP server
- Use a previously defined embedded SMTP server (if available)
- Define a new embedded SMTP server

If you choose to use the system’s global embedded SMTP server, the trading engine sets up the e-mail address for you.

If you choose to use a previously defined embedded SMTP server, the wizard prompts you to select the server.

If you choose to define a new embedded SMTP server, the wizard prompts for a server name and port number.

Once you have set up the transport, you might have to adjust some server settings. See Global embedded SMTP server on page 169 or Community embedded SMTP fields on page 173.



Configure the e-mail settings

Delivery exchange for receiving messages from partners

Community: *Worldwide Trading*

Complete the following fields for this transport server

Friendly name:*

(Example: My Server)

Port:* 4025

Figure 34. Add embedded e-mail server in delivery exchange wizard

The following fields are used in the delivery exchange wizard for adding an embedded SMTP server transport by defining a new embedded SMTP server.

Friendly name

A name identifying the embedded SMTP server. You can use any name you want.

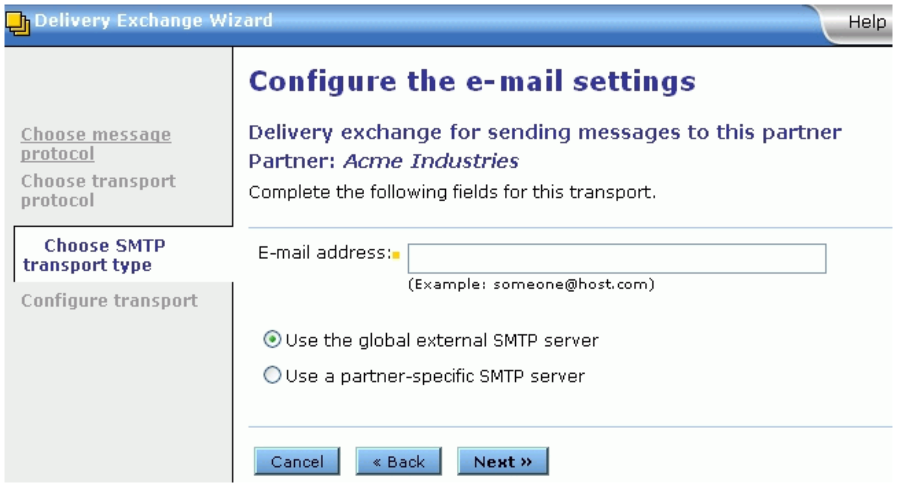
Port

The port number that listens for incoming SMTP connections. The default is 4025.

E-mail transport for partners

The configuration for using e-mail to send messages to partners provides the option of using the global external SMTP server or an external server only for use for a specific partner.

The following are fields for configuring the e-mail server transport for a partner in the delivery exchange wizard.



Configure the e-mail settings

Delivery exchange for sending messages to this partner
Partner: *Acme Industries*
Complete the following fields for this transport.

E-mail address:
(Example: someone@host.com)

(●) Use the global external SMTP server
( ) Use a partner-specific SMTP server

Figure 35. Configure partner e-mail (1 of 2)

The following three fields are shown in Figure 35.

E-mail address

The e-mail address for sending messages to a partner.

Use the global external SMTP server

Selecting this means the system's external SMTP server is used to send messages to the partner. The link to configure the system's external SMTP server is on the system management page of the user interface.

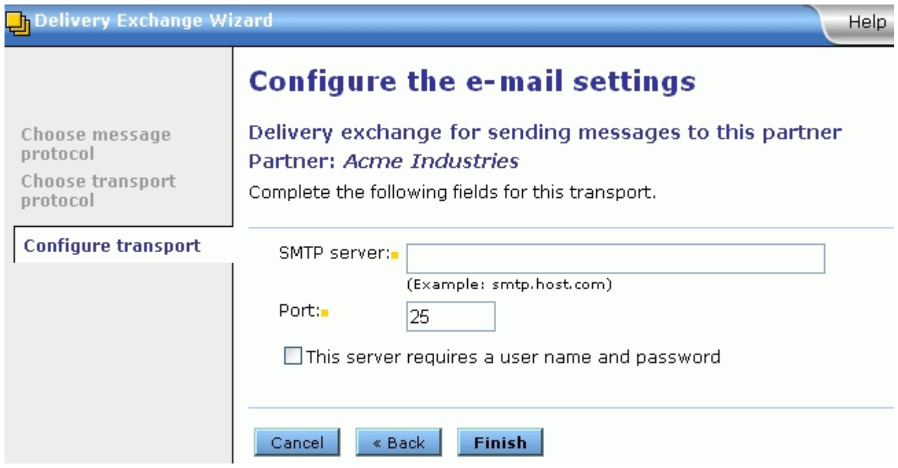
If you select this, click **Next** and the configuration is completed.

See Configuring external SMTP server on page 24.

Use a partner-specific SMTP server

Selecting this means you can specify an external SMTP server to send messages to the partner that is different from the system's external SMTP server.

If you select this, click **Next** and continue the configuration. See Figure 36.



Configure the e-mail settings

Delivery exchange for sending messages to this partner

Partner: *Acme Industries*

Complete the following fields for this transport.

SMTP server:

(Example: smtp.host.com)

Port: 25

☐ This server requires a user name and password

Figure 36. Configure partner e-mail (2 of 2)

SMTP server

Enter an SMTP server for sending messages just for to this partner. You must type a fully qualified domain name or IP address for the server. If you leave this field blank, the system inserts its external SMTP server.

Port

The default port for sending mail is 25.

This server requires a user name and password

Select this if a user name and password are required to connect to the server and then complete the following fields. SMTP servers usually do not require user names and passwords for sending.

User name

The user name to connect to the server.

Password

The password to connect to the server.

Confirm password

The password to connect to the server.

Embedded HTTP transport

A community can use an embedded HTTP server to receive messages from partners.

When you elect to set up an HTTP transport for a community using an embedded server, the delivery exchange wizard asks whether you want to:

- Use the system's global embedded HTTP server
- Use a previously defined embedded HTTP server (if available)
- Define a new embedded HTTP server

If you choose to use the system's global embedded SMTP server, the wizard prompts for a community routing ID to append to the URL. No other configuration is required.

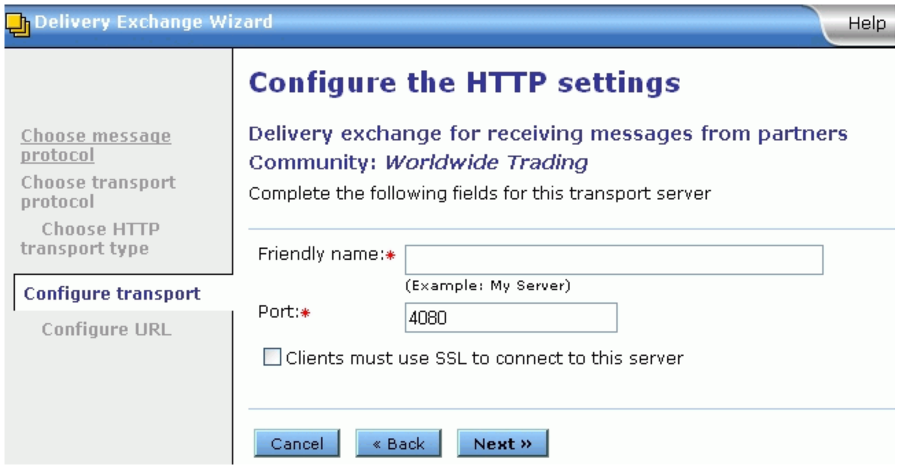
If you choose to use a previously defined embedded HTTP server, the wizard prompts for the server name and a community routing ID to append to the URL. No other configuration is required.

If you choose to define a new embedded HTTP server, the wizard prompts for a server name, port number and whether clients must use SSL to connect to the server.

If you choose to define a community-level enbedded HTTPS server, you must add an SSL certificate for the server. After setting up the server in the delivery exchange wizard, go to the maintenance page in the user interface for the server. Click **Change an embedded transport server** on the community summary page and click the name of the server. If the server needs a certificate, you are prompted to click **Add an SSL authentication certificate**. This action opens the wizard for adding a certificate.

Except for the global embedded server, embedded HTTP servers can be designated as HTTPS.

See Embedded transport servers on page 167 for information about changing the configurations of embedded servers.



Configure the HTTP settings

Delivery exchange for receiving messages from partners

Community: *Worldwide Trading*

Complete the following fields for this transport server

Friendly name:*

(Example: My Server)

Port:* 4080

☐ Clients must use SSL to connect to this server

Figure 37. Add embedded HTTP server in delivery exchange wizard (1 of 2)

The following fields are used in the delivery exchange wizard for adding an embedded HTTP server transport by defining a new embedded HTTP server.

Friendly name

Type a name for the new embedded HTTP server. This can be any name you want. You can select this sever when setting up additional embedded HTTP delivery exchanges later.

Port

The port number that listens for incoming HTTP connections. The default is 4080.

Clients must use SSL to connect to this server

Select this to set up an HTTPS delivery exchange. A clear box indicates HTTP.

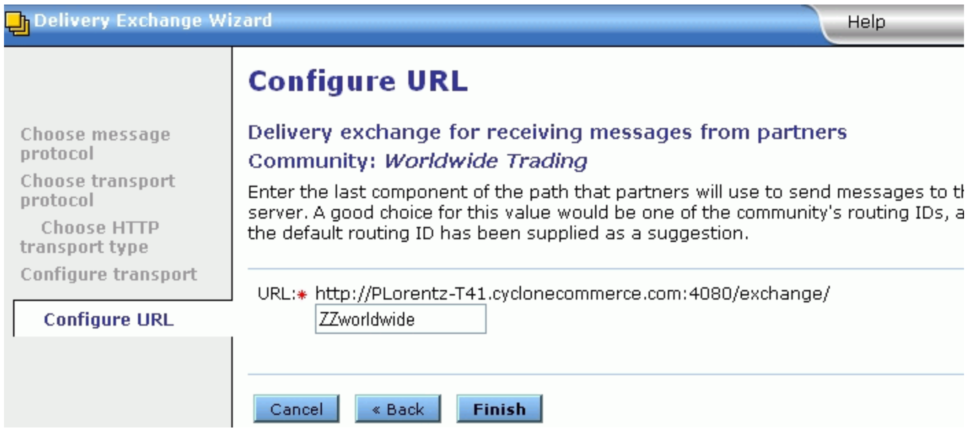
When you select this check box, the following sub-field displays.

This server requires client-side certificate authentication

If you selected SSL, select this check box to require your partners to submit a certificate to verify their identity before the delivery exchange allows the connection. Clear this box to use non-authenticated HTTPS.

If you select this, you must add an authentication certificate for the partner.

Click **Next** to continue the configuration. See Figure 38.



Configure URL

Delivery exchange for receiving messages from partners

Community: *Worldwide Trading*

Enter the last component of the path that partners will use to send messages to th server. A good choice for this value would be one of the community's routing IDs, a the default routing ID has been supplied as a suggestion.

URL:* http://PLorentz-T41.cyclonecommerce.com:4080/exchange/ ZZworldwide

Figure 38. Add embedded HTTP server in delivery exchange wizard (2 of 2)

URL

The wizard displays most of the URL for the transport, but you must provide a community routing ID to append to the end. You can use the ID the wizard displays or type another. This is the URL your partner uses to send messages to the community.

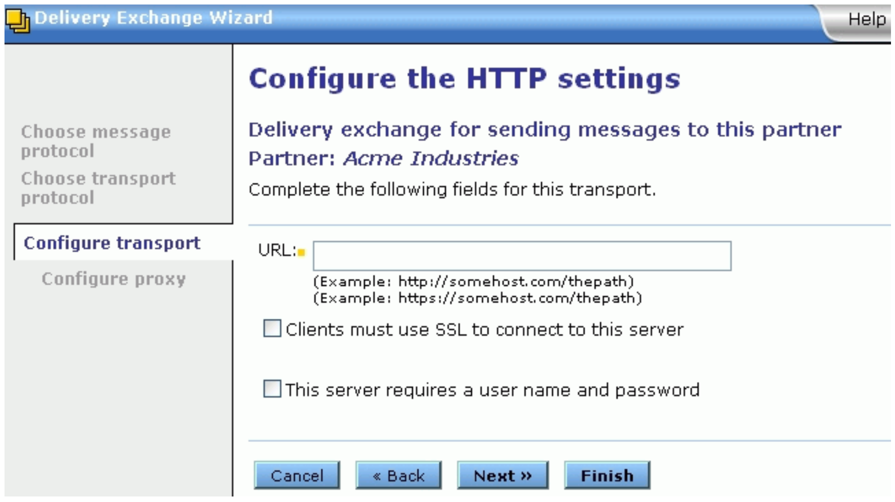
While this field is visible on the delivery exchange’s maintenance page, you can only edit it on the embedded server maintenance page. See Change embedded transports on page 302.

Staged HTTP transport

Setting up the staged HTTP transport requires installing and configuring a web servlet application in addition to using the delivery exchange wizard to add the transport. For details see Staged HTTP on page 235.

HTTP transport for partners

Communities can send messages to partners via an external HTTP server.



Configure the HTTP settings

Delivery exchange for sending messages to this partner

Partner: *Acme Industries*

Complete the following fields for this transport.

URL:

(Example: http://somehost.com/thepath)
(Example: https://somehost.com/thepath)

☐ Clients must use SSL to connect to this server

☐ This server requires a user name and password

Figure 39. Add HTTP transport for partner (1 of 2)

The following fields are used in the delivery exchange wizard for configuring this transport.

URL

The URL for connecting to the server.

Clients must use SSL to connect to this server

Select this to have Secure Sockets Layer protocol in use during connections. The server presents a certificate for verification. To do this, a certificate in a profile must be designated as the SSL certificate. The server must support SSL. If this is not selected, connections are anonymous.

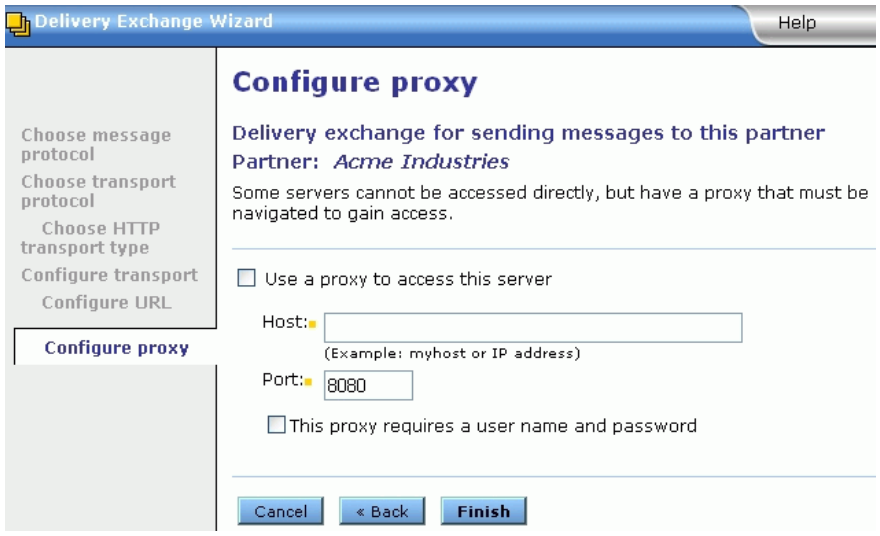
Enable host name verification

If selected, the trading engine compares the name of the SSL server to the name in the server's certificate to ensure they are the same.

This server requires a user name and password

If selected, type a user name and password for the community to connect to the server.

If the connection to the server will not made through a proxy, click **Finish**. If a proxy will be used, click **Next** and continue with the configuration. See Figure 40.



Configure proxy

Delivery exchange for sending messages to this partner

Partner: *Acme Industries*

Some servers cannot be accessed directly, but have a proxy that must be navigated to gain access.

☐ Use a proxy to access this server

Host:

(Example: myhost or IP address)

Port: 8080

☐ This proxy requires a user name and password

Figure 40. Add HTTP transport for partner (2 of 2)

Use a proxy to access this server

Select this if the connection to the partner must be made through a proxy. This normally is not required. Consult with your network administrator and your partner.

Host

The proxy host name.

Port

The proxy port.

This proxy requires a user name and password

If selected, type a user name and password for the community to connect to the proxy server.

File system transport

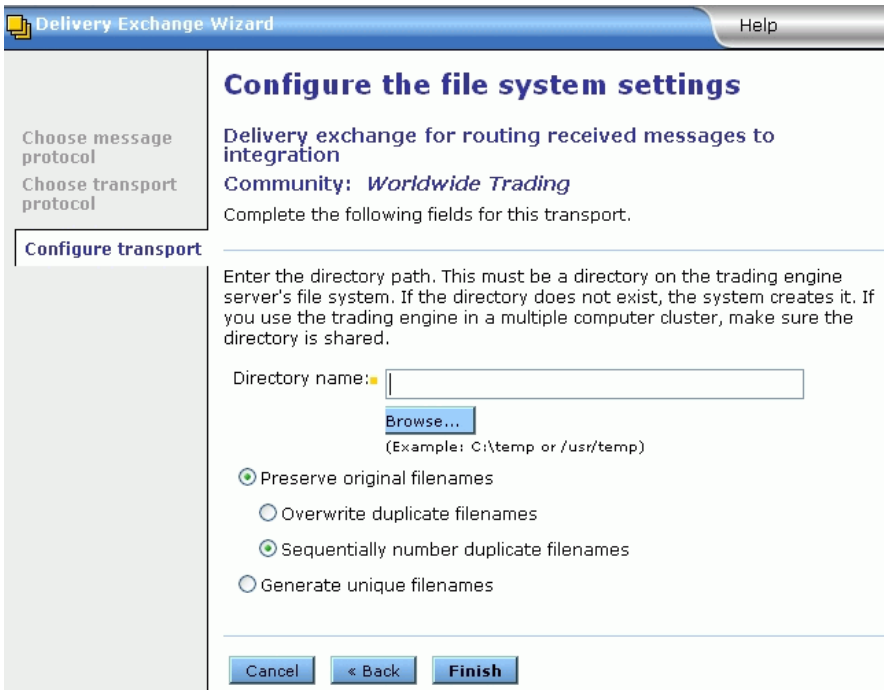
The file system transport can be used as a trading or integration exchange, but most often is used for integration.

Note: There is a quick way to set up directories for file system integration pickup and delivery. See Manage file system integration on page 227.

When you specify inbound and outbound file system integration directories, the system creates subdirectories for its own use. The system must have permissions to create the subdirectories. Do not delete these subdirectories or copy files to them or retrieve files from them.

For outbound integration, the system creates a subdirectory named **.ready**. After a message has been completely written to the parent outbound directory, the system moves it briefly to the ready subdirectory before consuming it. For inbound integration, the system creates a subdirectory named **.working**. After an inbound message has been completely written to the working directory, the system moves it to the inbound parent directory, where a back-end system can retrieve it.

Periods precede the names of the ready and working subdirectories. This is so the trading engine and any external applications can identify these as temporary directories and ignore them and any contents.



Configure the file system settings

Delivery exchange for routing received messages to integration

Community: *Worldwide Trading*

Complete the following fields for this transport.

Enter the directory path. This must be a directory on the trading engine server's file system. If the directory does not exist, the system creates it. If you use the trading engine in a multiple computer cluster, make sure the directory is shared.

Directory name:

(Example: C:\temp or /usr/temp)

[✓ Preserve original filenames]

Overwrite duplicate filenames

[✓ Sequentially number duplicate filenames]

Generate unique filenames

Figure 41. Add file system transport in delivery exchange wizard

The following fields are used in the delivery exchange wizard for configuring this transport.

Directory name

Use the **Browse** button or type the full path for a unique directory where the trading engine picks up or routes messages, depending on whether the transport is used for sending or receiving. If the directory does not exist, the trading engine creates it for you.

Use a unique directory name. You may want to give the directory a name that indicates whether the transport is being used for inbound or outbound integration, receiving messages from partners or sending messages to partners. For example, for outbound integration you could name the pickup directory \data\out; for inbound integration \data\in.

The following fields appear only when this transport is configured to send messages to partners or route messages received from partners to a back-end system. These fields do not apply to the file system with message meta-data transport for ebXML or RosettaNet.

Preserve original filenames

Select this if you want original file names to be preserved when the trading engine delivers messages.

For binary messages, we recommend that you preserve original file names. Otherwise, the trading engine assigns a unique file name that does not readily identify the contents of the file. Preserving original file names also allows your back-end application to process binary messages based on their file names.

This field applies to community integration delivery exchanges and partner delivery exchanges only.

Overwrite duplicate filenames

An option when you choose to preserve original file names. If duplicate file names are detected, the trading engine overwrites the existing file.

Sequentially number duplicate filenames

An option when you choose to preserve original file names. If duplicate file names are detected, the trading engine appends a number to the new file.

Generate unique filenames

Select to have the system provide a unique file name instead of using the original name.

This field applies to community integration delivery exchanges and partner delivery exchanges only.

When selected, files are given arbitrary names. The names always have less than 30 characters and often have less than 20 characters.

Appended to the file name is a hex representation of a monotonically increasing file name counter that is maintained in the database and guaranteed to be unique across all nodes in a cluster. In addition, if the original file name had an extension, the same extension is appended to the unique name the system generates.

The following are examples of unique file names generated by the system, one with the original file extension and one without:

dabeed45_4cb.edi
z47e4120_4ce

FTP transport

The FTP transport can be used as a trading or integration transport.

To enable partners to send messages to your FTP server, first set up the account, user ID and password for the FTP server where the trading engine retrieves files. Any partner who intends to receive messages from you by FTP also must also perform this step.

Note: We recommend using the AS3 message protocol rather than the secure file protocol to trade securely via FTP. An exception would be trading with a partner who uses an earlier version of Cyclone Interchange or Activator that does not support AS3. If you use Microsoft Internet Information Services (IIS) for the FTP server, make sure it is updated with the latest patches from Microsoft. Large files (approximately 1 gigabyte) may fail unless IIS is up to date.
See http://support.microsoft.com/?kbid=828086.

Information-level messages about FTP client-server interaction write to the trading engine log file. For more verbose messages to aid in troubleshooting FTP issues, you can change the message level to **debug**. Open the **log4j.properties** file in [install directory]\[build number]\conf. Search for the following entry, change the value from **info** to **debug**, and save and close the file.

```
log4j.category.com.cyclonecommerce.tradingengine.transport.ftp.SimpleD
ebug=info
```

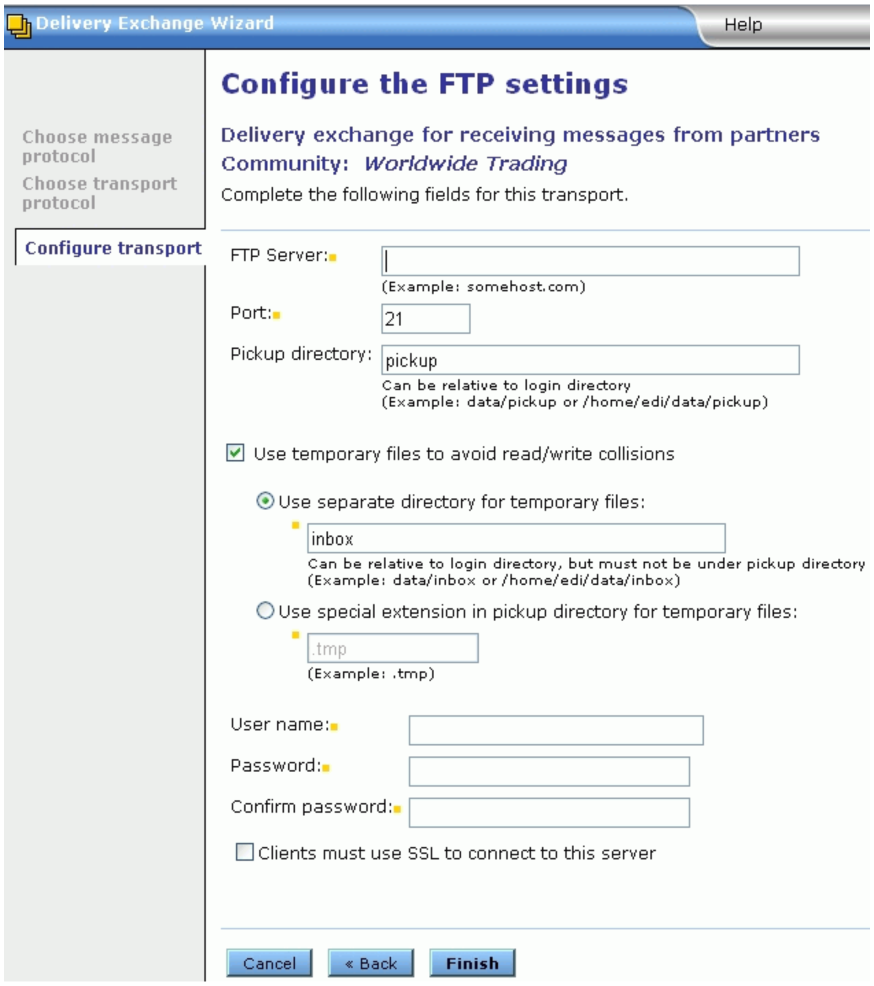
The debug setting logs each meta-command (for example, Send), followed by each primitive command as it is sent to the server (Rest, Stor, Rnfr, Rnto, and so on). Also, a message is logged each time a data connection is initiated or accepted. Each file name received during List operations is logged.

For more information see System logs on page 83.

When used for integration delivery or sending to partners, default settings are in place to preserve original file names and sequentially number duplicate file names. With these settings, the trading engine uses a file naming convention to defeat conflicts arising from multiple nodes feeding identically named documents to the same FTP directory and server. Files are renamed in the following format:

[original file name]__[unique message ID].[original file extension]

For information about how to test FTP connections see FTP tester tool on page 429.



Configure the FTP settings

Delivery exchange for receiving messages from partners
Community: *Worldwide Trading*
Complete the following fields for this transport.

FTP Server:
(Example: somehost.com)
Port: 21
Pickup directory: pickup
Can be relative to login directory
(Example: data/pickup or /home/edi/data/pickup)

☑ Use temporary files to avoid read/write collisions

● Use separate directory for temporary files:
inbox
Can be relative to login directory, but must not be under pickup directory
(Example: data/inbox or /home/edi/data/inbox)

○ Use special extension in pickup directory for temporary files:
.tmp
(Example: .tmp)

User name:
Password:
Confirm password:

☐ Clients must use SSL to connect to this server

Figure 42. Add FTP transport in delivery exchange wizard

The following fields are used in the delivery exchange wizard for configuring this transport.

FTP Server

The name of the FTP server.

Port

The port on which the server listens for incoming connections. The default is 21.

Pickup directory

Type the path of the directory on your server where messages are picked up. When the trading engine polls the server for files, it only looks in the pickup directory, not an inbox directory.

Use temporary files to avoid read/write collisions

We recommend using this option to prevent the trading engine from attempting to retrieve partially written files. When this is selected, you must select one of the two following options.

There may be some specialized servers, typically running on mainframes, that support only part of the FTP protocol (RFC 959). In such cases you may have to clear this check box and take steps of your own to make sure collisions do not occur.

Use separate directory for temporary files

Type the full path of an inbox directory (for example, c:\data\inbox). Files are uploaded to this directory. When fully written, files are moved to the pickup directory for retrieval.

Do not put the inbox under the pickup directory unless you use a period at the beginning of the inbox name. The trading engine and other applications ignore directories and files that begin with periods.

For example, do not use the following directory structure:

c:\data\pickup\inbox

But you can use the following because a period is the first character of the inbox directory name:

c:\data\pickup\.inbox

When receiving files from a partner, we recommend that your partner write files to the inbox directory first and then move them to the pickup directory when they are ready to be retrieved. This process is automatic if your partner also uses Cyclone Interchange or Activator. If the partner uses other software to upload files to your server, the software should be configured to initially upload the files to the inbox directory and move them to the pickup directory when they are ready to be retrieved.

For outbound integration, the back-end system must write the message to the inbox and then move it to the pickup directory.

For inbound integration and sending outbound to partners, the trading engine writes to the inbox and then moves the message to the pickup directory.

Use special extension in pickup directory for temporary files

If you prefer not to use an inbox, select this option. While a file is being written to the pickup directory, a temporary extension is added so the system knows not to retrieve it because the file is only partially written. Once fully written, the temporary extension goes away and the file can be retrieved.

User name

The user name to connect to the server.

Password

The password to connect to the server.

Confirm password

The password to connect to the server.

Clients must use SSL to connect to this server

Select this to have Secure Sockets Layer protocol in use during connections. The server presents a certificate for verification. To do this, a certificate in a profile must be designated as the SSL certificate. The server must support SSL. If this is not selected, connections are anonymous.

Enable host name verification

If selected, the trading engine compares the name of the SSL server to the name in the server’s certificate to ensure they are the same.

SFTP transport

The Secure FTP (SFTP) transport can be used as a trading or integration transport.

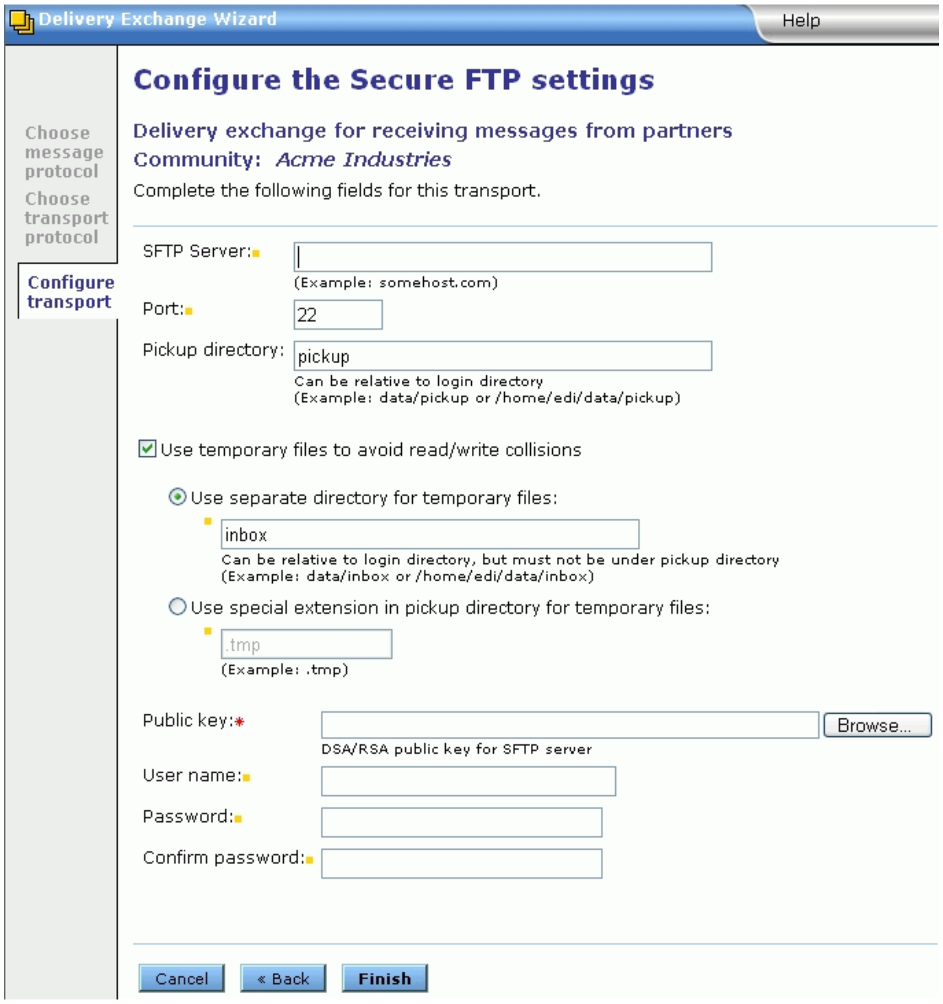
To enable partners to send messages to your SFTP server, first set up the account, user ID and password for the SFTP server where the trading engine retrieves files. Any partner who intends to receive messages from you by SFTP also must also perform this step.

SFTP is similar to FTP, but performs all operations over an encrypted Secure Shell (SSH) transport. SFTP and FTP/SSL (or FTPS) are different transports. An SFTP server can communicate only with other SFTP servers, not FTP servers.

The trading engine supports limited SFTP functionality as the following notes:

- Only supports SSH 2.0 (see http://www.ietf.org/html.charters/secsh-charter.html or http://www.openssh.com/txt/draft-ietf-secsh-filexfer-02.txt).
- Only supports user name and password authentication.
- Checkpoint-restart functionality is not supported.
- User commands and scripting (as supported for FTP) are not supported for SFTP.

This transport has been tested only with the OpenSSH sftp-server with user name and password authentication.


Delivery Exchange Wizard

Help

Choose message protocol

Choose transport protocol

Configure transport

Configure the Secure FTP settings

Delivery exchange for receiving messages from partners

Community: *Acme Industries*

Complete the following fields for this transport.

SFTP Server:
(Example: somehost.com)

Port: 22

Pickup directory: pickup
Can be relative to login directory
(Example: data/pickup or /home/edi/data/pickup)

☑ Use temporary files to avoid read/write collisions

● Use separate directory for temporary files:
inbox
Can be relative to login directory, but must not be under pickup directory
(Example: data/inbox or /home/edi/data/inbox)

○ Use special extension in pickup directory for temporary files:
.tmp
(Example: .tmp)

Public key:* Browse...
DSA/RSA public key for SFTP server

User name:

Password:

Confirm password:

Cancel « Back Finish

Figure 43. Add SFTP transport in delivery exchange wizard

SFTP fields

The following fields are used in the delivery exchange wizard for configuring this transport.

SFTP Server

The name of the SFTP server.

Port

The port on which the server listens for incoming connections. The default is 22.

Pickup directory

Type the path of the directory on your server where messages are picked up. When the trading engine polls the server for files, it only looks in the pickup directory, not an inbox directory.

Use temporary files to avoid read/write collisions

We recommend using this option to prevent the trading engine from attempting to retrieve partially written files. When this is selected, you must select one of the two following options.

Use separate directory for temporary files

Type the full path of an inbox directory (for example, c:\data\inbox). Files are uploaded to this directory. When fully written, files are moved to the pickup directory for retrieval.

Do not put the inbox under the pickup directory unless you use a period at the beginning of the inbox name. The trading engine and other applications ignore directories and files that begin with periods.

For example, do not use the following directory structure:

c:\data\pickup\inbox

But you can use the following because a period is the first character of the inbox directory name:

c:\data\pickup\.inbox

When receiving files from a partner, we recommend that your partner write files to the inbox directory first and then move them to the pickup directory when they are ready to be retrieved. This process is automatic if your partner also uses Cyclone Interchange or Activator. If the partner uses other software to upload files to your server, the software should be configured to initially upload the files to the inbox directory and move them to the pickup directory when they are ready to be retrieved.

For outbound integration, the back-end system must write the message to the inbox and then move it to the pickup directory.

For inbound integration and sending outbound to partners, the trading engine writes to the inbox and then moves the message to the pickup directory.

Use special extension in pickup directory for temporary files

If you prefer not to use an inbox, select this option. While a file is being written to the pickup directory, a temporary extension is added so the system knows not to retrieve it because the file is only partially written. Once fully written, the temporary extension goes away and the file can be retrieved.

Public key

The path to the file that contains the RSA or DSA public key for the SFTP server. You must get this file from the server administrator. The trading engine uses the key to authenticate the server.

User name

The user name to connect to the server.

Password

The password to connect to the server.

Confirm password

The password to connect to the server.

Testing SFTP

You can use the sftpTester tool to exercise the SFTP client outside of the trading engine. The script to start sftpTester can be found in [install directory]\[build number]\tools.

sftpTester is a console-based application that can verify the operation of the SFTP client in the trading engine and a partner's SFTP server. The trading engine server does not have to be running to use this tool. You can use it on UNIX or Windows.

sftpTester duplicates the way the trading engine accesses an SFTP server. It is a test program to verify interoperability with SFTP servers. If you can send, list, receive and delete files on a SFTP server using sftpTester, this is a good indication the trading engine can interoperate with the server.

sftpTester prompts for all the information it needs, as the following illustrates:

```
**** Welcome to the Cyclone Sftp test program ****
-> Enter host: localhost
-> Enter user: ftpuser
-> Enter password: ftpuserpwd
-> Enter C for CONSUMER client (list, receive, delete), P for PRODUCER
(send). (Blank assumes C):
```

```
-> Enter pickup directory (blank for "pickup"):
-> Enter public key path/filename: ssh_host_dsa_key.pub
```

After prompting for the initial configuration information such as the host, user and password, sftpTester displays a main prompt that allows you to enter meta-commands to perform simple operations such as list, send and receive. You can enter a question mark (?) at this point to get more information. The following information displays upon entering a question mark at the main prompt:

Consumer commands

```
-> Enter CONSUMER command (e.g. ?, LIST, RECeive, DELete, LLIST, LCD,
QUIT): ?
CONSUMER metacommands (abbreviations shown in upper case):
?                  help
LIST               list files on host
RECeive filename   retrieve file from host
DELete filename    delete file from host
LCD                change local working directory
LLIST              list files in local working directory
QUIT/EXIT/BYE      exitNormal SftpTester
```

Producer commands

```
-> Enter PRODUCER command (e.g. ?, SEND, LLIST, LCD, QUIT): ?
PRODUCER metacommands (abbreviations shown in upper case):
?                  help
SEND filename      write file to host
LCD                change local working directory
LLIST              list files in local working directory
QUIT/EXIT/BYE      exitNormal SftpTester
```

Troubleshooting SFTP

For troubleshooting, you can write messages specific to the SFTP transport to the trading engine log file. You can add the following properties to the log4j.properties file at [install directory]\[build number]\conf.

- For messages related to high-level operation of the SFTP client, this property in debug mode is useful for finding common SFTP problems.

  log4j.category.com.cyclonecommerce.tradingengine.transport.sftp.SimpleDebug=debug

- For messages related to low-level operation of the SFTP client, this property in debug mode produces verbose messages. (Try the simple debug property before using this one.)

  log4j.category.com.cyclonecommerce.tradingengine.transport.sftp=debug

JMS transport

The Java Message Service (JMS) transport normally is used as an integration transport, but can be used as a trading transport.

To use this transport your community must have JMS experience and a working JMS system.

Your JMS system may have file size limitations. Check the documentation for your JMS system.

The JMS integration transport is an input and output source for messages. This is how it works: the trading engine polls the JMS server for messages in the designated inbound queue. This means that any JMSMessage placed in the queue by another process is retrieved by the trading engine, which verifies the type of JMSMessage (BytesMessage or TextMessage). If verified, the trading engine packages and sends it to the partner. Likewise, every message the trading engine receives from a partner is unpackaged, converted to a BytesMessage or TextMessage and placed on the designated outbound queue.

For pickup integration exchanges, the trading engine determines whether the message read from the queue is a BytesMessage or a TextMessage. For delivery integration exchanges, in which messages from partners are sent to back-end systems, you can select the JMS type (BytesMessage or TextMessage) on the Advanced tab of the transport's maintenance page.

When using JMS for integration, configure the trading engine to parse EDI and XML messages retrieved from a back-end system for sender and receiver information.

Binary documents retrieved from the back-end must have the following meta-data string parameters appended to the BytesMessage or TextMessage: SenderRoutingId, ReceiverRoutingId and ContentMimeType. The trading engine performs routing decisions based on the string parameters.

When the trading engine sends a BytesMessage or TextMessage to integration, it includes the string parameters SenderRoutingId, ReceiverRoutingId and ContentMimeType for all document types.

Note that when the trading engine encounters a meta-data element containing characters that JMS cannot recognize, it changes the offending characters into the hex representation of the ASCII code of the characters. For example, the meta-data element **ediint.DocumentType** becomes **ediint$2eDocumentType**. The **$2e** is the hex representation of a period. When submitting JMS messages to the trading engine, use the properly encoded hex names to have them turned into the proper meta-data names.

In addition to using the delivery exchange wizard to configure a JMS transport, other set up is required. Depending on your provider, follow the recommendations in the Most providers or Oracle AQ topic.

Note: If BEA WebLogic is your JMS provider, there are options for ensuring proper load balancing and fail-over when delivering messages to clustered JMS servers. One option is to configure a distributed destination in WLS and reference its JNDI name on the JMS configuration page in Cyclone Activator. At runtime, the JNDI lookup performed by the WebLogic JMS client resolves the distributed destination name to a physical queue. Another option is to provide a comma-separated list of host names in the JNDI URL field in Cyclone Activator (for example, t3://node1:7001,node2:7002,node3:7003). At runtime, the JMS client round-robins between the specified providers. Both options ensure load balancing and support for fail-over. See the WebLogic documentation for how to configure these options.

Most providers

In addition to completing the JMS transport configuration page in the user interface, you must add the name of the JAR or class files or both in the environment file so the server can locate the JMS and JNDI provider. The environment file is in [install directory]\[build number]\ bin. In some cases, you need to add the path and name of only one JAR file (for example, swiftmq.jar or weblogic.jar), but you might have to include a series of jars or paths. Do not put the JAR in the application's corelib directory.

In Windows, add the JAR at the end of the class path in the following lines in the environment file:

```
set CYC_CP=..\classes;..\jars;..\corelib
set
CYC_CP=%CYC_CP%;..\corelib\db\sqlserver;..\corelib\db\oracle
;..\corelib\db\db2;..\corelib\db\derby
```

For example, for WebLogic on Windows, the entry would be similar to the following:

```
set CYC_CP=..\classes;..\jars;..\corelib
set
CYC_CP=%CYC_CP%;..\corelib\db\sqlserver;..\corelib\db\oracle
;..\corelib\db\db2;..\corelib\db\derby;
C:\bea\weblogic81\server\lib\weblogic.jar
```

If you run on Windows and use the Windows service for starting the trading engine server, also add the JMS JAR to the end of the following class path line in the CycloneService.ini file, which also is in the bin directory.

```
CYC_CP=..\classes;..\jars;..\corelib;..\corelib\db\sqlserver
;..\corelib\db\oracle;..\corelib\db\db2;..\corelib\db\derby
```

In UNIX, locate the following lines in the environment file:

```
# Set classpath used by Cyclone class loader (will load all
jars in directories).
CYC_CP="${CYC_DIR}/classes"
CYC_CP="${CYC_CP}:${JARS}"
CYC_CP="${CYC_CP}:${CYC_DIR}/corelib"
CYC_CP="${CYC_CP}:${CYC_DIR}/corelib/db/sqlserver"
CYC_CP="${CYC_CP}:${CYC_DIR}/corelib/db/oracle"
CYC_CP="${CYC_CP}:${CYC_DIR}/corelib/db/db2"
CYC_CP="${CYC_CP}:${CYC_DIR}/corelib/db/derby"
```

After the last line, insert a line specifying the JAR location. For example:

```
CYC_CP="${CYC_CP}:/bea/weblogic81/server/lib/weblogic.jar"
```

Oracle AQ

If the provider is Oracle Advanced Queuing facility (Oracle AQ), Oracle must be the external database for the trading engine and Oracle AQ must be installed.

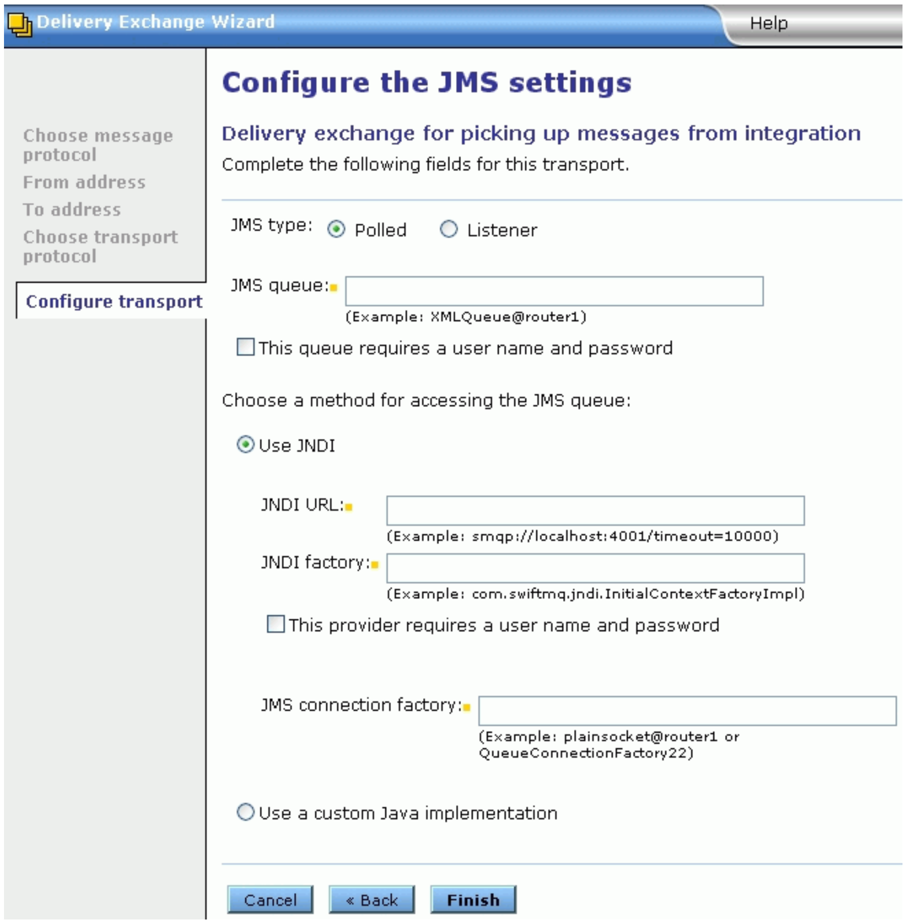
Add a message queue on Oracle AQ. The queue payload type must be one of the following:

SYS.AQ$_JMS_BYTES_MESSAGE

SYS.AQ$_JMS_TEXT_MESSAGE

On your Oracle client, copy the Oracle AQ drivers **aqapi.jar** and **jmscommon.jar**. You may find these files in the rdbs/jlib directory. Paste the files to the Cyclone Activator directory [install directory]\[build number]\corelib\db\oracle and restart the server.

For any JMS transport for Oracle AQ that polls for messages, go to the Advanced tab on the transport's maintenance page and select **Use transacted queue**. You need to do this if the JMS settings tab name includes the word **polled**. For example, **JMS (polled) settings**. If the settings tab is named **JMS (listener) settings**, the **Use transacted queue** control is not available on the Advanced tab.



Configure the JMS settings

Delivery exchange for picking up messages from integration

Complete the following fields for this transport.

JMS type: [✓ Polled] Listener

JMS queue:

(Example: XMLQueue@router1)

☐ This queue requires a user name and password

Choose a method for accessing the JMS queue:

[✓ Use JNDI]

JNDI URL:

(Example: smqp://localhost:4001/timeout=10000)

JNDI factory:

(Example: com.swiftmq.jndi.InitialContextFactoryImpl)

☐ This provider requires a user name and password

JMS connection factory:

(Example: plainsocket@router1 or QueueConnectionFactory22)

Use a custom Java implementation

Figure 44. Add JMS transport in delivery exchange wizard

JMS fields

The following fields are used in the delivery exchange wizard for configuring this transport.

Except for the user name and password, you can obtain the information needed to complete this page from the JMS provider's documentation. The information varies depending on the provider. If you have questions, contact your JMS provider.

JMS type

There are two choices for acquiring messages from a JMS queue for integration pickup and receiving messages from partners:

Polled. The trading engine polls the JMS server at intervals for messages. This is the default setting. The polling interval and the maximum number of files to retrieve per interval can be adjusted on the Advanced tab of the transport's maintenance page. Although the rate at which messages enter the system is controlled, this selection introduces latency and overhead in checking the JMS server when the queue is empty.

Listener. The JMS server calls the trading engine as soon as a message is written to its queue. Messages are transferred as they arrive and do not wait for the trading engine to poll for them. When selected, the transport's Advanced tab on the maintenance page has a setting for reconnecting to the JMS server if the server goes down. However, there are no controls related to polling, because polling does not apply. Although latency and polling overhead are eliminated, this selection cannot control the rate at which messages enter the system, which could overload it.

Once polled or listener has been selected, the JMS type cannot be changed on the transport's maintenance page.

JMS queue

The name of the queue. Example: XMLQueue@router1

This queue requires a user name and password

Select this if the queue requires a user name and password. When selected, fields appear for entering the information.

User name

A user name for the JNDI provider. This value could be blank and is typically provided for in the JNDI URL. However, this depends on the JNDI provider and how it is configured.

Password

A password for the JNDI provider. This value could be blank and is typically provided in the JNDI URL. However, this depends on the JNDI provider and how it is configured.

Confirm password

Type the password again for confirmation.

Use JNDI

Select this if a Java Naming and Directory Interface (JNDI) provider. When selected the applicable fields display.

JNDI URL

The network URL used to obtain access to the JNDI service provider for your JMS service. Example: smqp://localhost:4001/timeout=10000

JNDI factory

The name for the JNDI service provider class. Example: com.swiftmq.jndi.InitialContextFactoryImpl

This provider requires a user name and password

Select this if JMS requires a user name and password. When selected, fields appear for entering the information.

User name

A user name for the JMS provider. This can be the same as your JNDI user name. However, this depends on your JMS provider and how it is configured.

Password

A password for the JMS provider. This can be the same as your JNDI password. However, this depends on your JMS provider and how it is configured.

Confirm password

The password again for confirmation.

JMS connection factory

The connection factory as defined within the JMS provider. This value can be either in the form **factoryname@routername** or the JNDI public symbol for the QueueConnectionFactory. Examples: plainsocket@router1 or QueueConnectionFactory22. This depends on your JMS provider and how it is configured.

Use a custom Java implementation

Select this for a JMS provider that does not require a JNDI implementation. For example, Oracle Advanced Queuing facility (Oracle AQ). When selected the applicable fields display.

Class name

The name of the Java class for implementing the connection to the message queue. A Java class for Oracle AQ is available. The class name is:

```
com.cyclonecommerce.tradingengine.transport.jms.OracleAQ
QueueUtil
```

If you want a Java class for a provider other than Oracle AQ, you need the help of a professional services consultant. Contact technical support for information.

Parameters

These are the parameters for implementing the Java class. There are four parameters required for the Java class for Oracle AQ. These parameters must be in the following order:

Host. The name of the computer running Oracle AQ.

Queue name. The name of the Oracle AQ message queue.

Port. The port Oracle AQ uses to listen for messages.

Driver type. The type of JDBC driver for connecting to the provider. For Oracle AQ, the valid values are **thin** and **oci8**.

Testing JMS

You can use the jmsTester tool to exercise the JMS client outside of the trading engine. The script to start jmsTester can be found in [install directory]\[build number]\tools.

jmsTester is a console-based application that can verify the operation of the JMS client in the trading engine and a partner's JMS server. The trading engine server does not have to be running to use this tool. You can use it on UNIX or Windows.

jmsTester duplicates the way the trading engine accesses a JMS server. It is a test program to verify interoperability with JMS servers. If you can send, list, receive and delete files on a JMS server using jmsTester, this is a good indication the trading engine can interoperate with the server.

jmsTester prompts for all the information it needs, as the following illustrates:

```
**** Welcome to the Cyclone JMS test program ****
-> (n)ew client, (c)onnect, (d)isconnect, (s)end, (r)etrieve,
(a)cknowledge, (q)
```

```
uit: n
-> Use JNDI (Y/N - default is yes):
-> JNDI URL (): url
-> JNDI Factory (): factory
-> JNDI Username (): name
-> JNDI Password (): pwd
-> JMS Queue Connection Factory(null): :
-> JMS Queue (null):
-> JMS Username (null):
-> JMS Password (null):
-> Use Transacted Queue (Y/N - default is no):
-> Send using Bytes or Text Message type: (bytes):
```

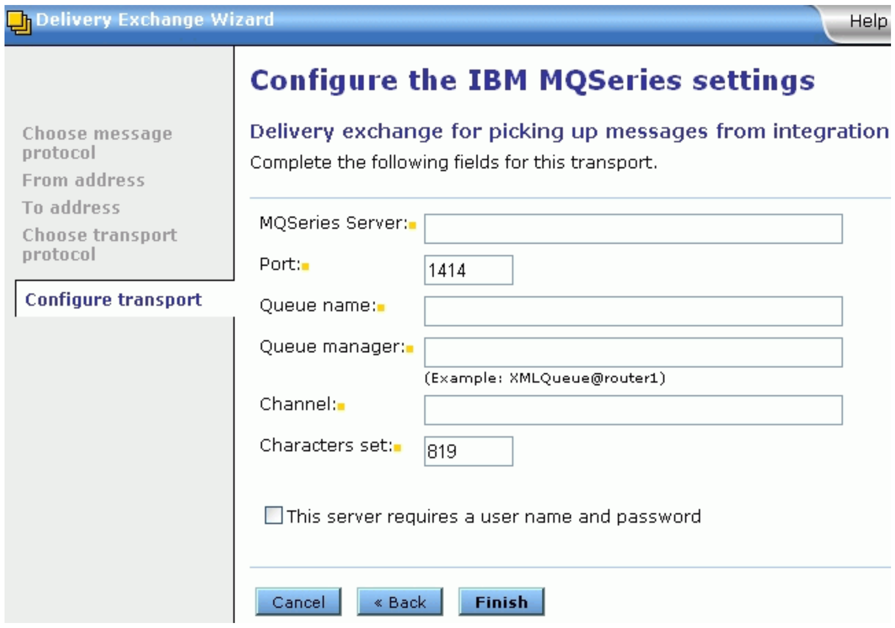
After prompting for the initial configuration information, you can use one of the testing commands, which are:

(c)onnect
(d)isconnect
(s)end
(r)etrieve
(a)cknowledge

IBM MQSeries transport

The MQSeries transport can be used as a trading or integration transport.

To use this transport, the MQSeries server, queue manager and queues should be properly configured.



Configure the IBM MQSeries settings

Delivery exchange for picking up messages from integration

Complete the following fields for this transport.

MQSeries Server:
Port: 1414
Queue name:
Queue manager:
(Example: XMLQueue@router1)
Channel:
Characters set: 819

☐ This server requires a user name and password

Figure 45. Add IBM MQSeries transport in delivery exchange wizard

The following fields are used in the delivery exchange wizard for configuring this transport.

MQSeries server

The fully qualified domain name or IP address of the MQSeries host.

Port

The port where the application listens for incoming documents. The default is 1414.

Queue name

The name of the MQSeries queue that receives incoming documents.

Queue manager

The name of the MQSeries queue manager.

Channel

The name of the communications channel.

Characters set

The character set used by the queue manager. This number should match the number used by the queue manager. The default is 819.

This server requires a user name and password

Select this check box to supply a user name and password for basic authentication. Clear this check box if the server does not require basic authentication.

User name

The user name to connect to the server.

Password

The password to connect to the server.

Confirm password

The password entered again for confirmation.

Web services API transport

The web services API transport can be used for integration only.

Integration pickup

When a community uses the web services API to retrieve messages from a back-end system, this transport uses a global embedded web services API server. The back-end posts messages to a URL in the following format:

http://localhost:5080/services/MessageService

The variable **localhost** is the fully qualified domain name or IP address of the computer running the trading engine. The default port is 5080, but this can be changed.

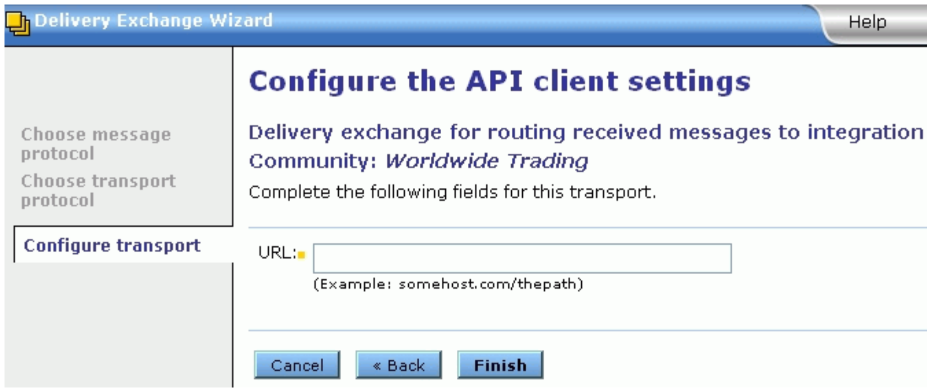
No user configuration is needed to set up this pickup transport for a community. You can modify the global embedded web services API server, however, by clicking **Configure the global embedded Web services API server** on the main Trading configuration page. Because the global server is shared by all communities, any configuration changes affect all.

Integration delivery

When a community uses the web services API to send messages received from partners to a back-end system, this transport uses an API client to send messages to a back-end web server. The set up requires the help of a technician or developer who is familiar with web services and the Web Services Description Language (WSDL). The technician or developer needs to provide a server implementation of the provided MessageService WSDL.

The WSDL that describes the requirements for both the server and client is at [install directory]\[build number]\conf\MessageService.wsdl.

Once the back-end web server has been configured according to the WSDL requirements, you can enter the URL for posting messages to the back-end server in the single configuration field for the web services API client in the delivery exchange wizard (Figure 46).



Configure the API client settings

Delivery exchange for routing received messages to integration

Community: *Worldwide Trading*

Complete the following fields for this transport.

URL:

(Example: somehost.com/thepath)

Figure 46. Add API client transport in delivery exchange wizard

After the transport has been set up, you have the option of sending the message payload (the default) or only the URL that points to the payload. If you choose to send only the URL, the back-end system uses the URL to retrieve the payload from the trading engine backup directory. To enable this option, click **Integration delivery** in the navigation graphic at the top of the community summary page and click the web services API client transport. Click the **Advanced** tab and select **Send the payload URL only**.

Because the payload is retrieved from the backup directory, there are two conditions that must be met if you choose to send the payload URL only:

- Backing up files must be enabled for the web services API client integration delivery exchange.
- If you have set up a schedule on the trading engine for deleting messages, the back-end must retrieve the payload before the next scheduled purge occurs or the payload will be lost. For information about setting up schedules for deleting messages see Data backups and deletes on page 421.

Post-processing configuration details

You can perform post-processing commands on each inbound message immediately after the trading engine has received, processed and written it to a delivery integration exchange. You also can execute post-processing commands after sending messages to partners. The trading engine can initiate any executable or batch file or script you specify. The batch file or script is applied to all messages passed through the exchange that calls it. Batch files or scripts can be used with several or all integration delivery exchanges or partner delivery exchanges, but they must be called separately. You cannot specify a global batch file or script.

The post-processing script must be on a drive that the trading engine can access and has permission to execute.

There are two places in the user interface where you can enter the name of a post-processing script.

- On a community summary page, click **Integration delivery** on the navigation graphic at the top of the page. Then click the name of an integration exchange to open the maintenance page. Click the **Advanced** tab. Type the script name in the post-processing script field. Click **Save changes**.
- On a partner summary page, click **Delivery exchange** on the navigation graphic at the top of the page. Then click the name of an outbound transport to open the maintenance page. Click the **Advanced** tab. Type the script name in the post-processing script field. Click **Save changes**.

The trading engine by default passes seven items of message meta-data to the post process. Your script can use any or all of them. An example of the syntax of a command that is executed against an inbound message is:

Windows	c:\directoryname\myfile.bat
UNIX	/directoryname/myscript.sh

Post-processing message meta-data

The default message meta-data for post-processing are described in the following table. These match the default post-processing meta-data elements in the shellscriptmetadata.xml file in [install directory]\[build number]\conf. The trading engine checks this file to determine valid meta-data for post-processing. The parameter numbers are shown for Windows and UNIX.

Windows	UNIX	Message meta-data	Description
%1	$1	SenderRoutingId	The routing ID of the sender of the message.
%2	$2	ReceiverRoutingId	The routing ID of the receiver of the message.

Windows	UNIX	Message meta-data	Description
%3	$3	ProductionUrl	The path of the file if the message was written to a File System integration delivery exchange. Otherwise, it is the URL of the destination.
%4	$4	ProductionFilename	The file name used when the trading engine wrote the file.
%5	$5	ConsumptionFilename	The file name included in the MIME header, probably the original file name from the sender, if the message was EDIINT. Otherwise, the name of the file as when retrieved from the file system or FTP server.
%6	$6	EdiControlId	The control ID of an EDI message. Otherwise, the ID is XML or BINARY
%7	$7	DocumentClass	The document class of the message payload (for example, X12, XML).

If you are writing a post-processing script in Perl, you cannot use $ in the parameter number. You must use $ARGV[n], where [n] is the argument number. For example, in the preceding table, the parameter for SenderRoutingId is $1; for Perl it is $ARGV0 (notice counting starts at zero). Likewise, the parameter for ReceiverRoutingId is $2, but $ARGV1 for Perl.

Adding post-processing elements

You can use other post-processing elements than those listed in the table in Post-processing message meta-data on page 223.

Add the meta-data elements you want to shellscriptmetadata.xml in [install directory]\[build number]\conf. Follow the format as shown for the default elements already in the file. Pay attention to the order in which the meta-data elements are listed. This is important for the corresponding parameter numbers.

After editing shellscriptmetadata.xml, restart the trading engine server for the changes to become effective. All post-processing script invocations are affected by changes to this file.

Methods for writing scripts

You can use the following methods for writing post-processing scripts:

Operating system	Languages
Windows	Compiled languages (for example, Java, Visual BASIC, C++, Delphi). Also, .cmd and .bat files.
UNIX	Any language. For example, shell script, Java, C or Perl.

Script example for Windows

The following is an example of a post-processing script for Windows. This script re-directs an inbound file to a local directory and writes activity to an external log file. This example is provided solely to illustrate the correct format for such scripts.

```
@echo off
rem WindowsPostprocess.bat to test post-processing.
rem This batch file does two things. It moves the received file
rem to another directory. Then it appends into a log file all
rem the information that CI makes available about that file.

@echo off
move %3\%4 c:\tmp
echo. >> c:\tmp\postprocess.log
echo -----newfile info----- >> c:\tmp\postprocess.log
date/t >> c:\tmp\postprocess.log
time/t >> c:\tmp\postprocess.log
echo The Sender Routing Id is %1 >> c:\tmp\postprocess.log
echo The Receiver Routing Id is %2 >> c:\tmp\postprocess.log
echo The Production Directory is %3 >> c:\tmp\postprocess.log
echo The Production Filename is %4 >> c:\tmp\postprocess.log
echo The Consumption Filename is %5 >> c:\tmp\postprocess.log
echo The Control Id is %6 >> c:\tmp\postprocess.log
echo The Content MIME Type is %7 >> c:\tmp\postprocess.log
```

Script example for UNIX

The following is an example of a post-processing script for UNIX. This script re-directs an inbound file to a local directory and writes activity to an external log file. This example is shown solely to illustrate the correct format for such scripts.

```
#!/bin/sh
# $Id: UNIXpostprocess.sh to test post-processing.
# This shell script does two things. It moves the received file
# to another directory. Then it appends into a log file all the
# information that CI makes available about that file.

mv "$3"/"$4" /home/cyclone/tmp
echo -----newfile info----- >> /home/cyclone/tmp/postprocess.log
date >> /home/cyclone/tmp/postprocess.log
echo The Sender Routing Id is "$1" >> /home/cyclone/tmp/postprocess.log
echo The Receiver Routing Id is "$2" >> /home/cyclone/tmp/postprocess.log
echo The Production directory is "$3" >> /home/cyclone/tmp/postprocess.log
echo The Production Filename is "$4" >> /home/cyclone/tmp/postprocess.log
echo The Consumption Filename  is "$5" >> /home/cyclone/tmp/postprocess.log
echo The Control Id is "$6" >> /home/cyclone/tmp/postprocess.log
echo The Content MIME Type is "$7" >> /home/cyclone/tmp/postprocess.log
```

Post-processing events

The trading engine generates the following events when performing post processing.

Messaging_Transport_PostProcessing_Initiated

This event is published before the script is invoked.

Messaging_Transport_PostProcessing_Completed

This event is published after the script has been invoked.

Messaging_Transport_PostProcessing_Failure

This event is published if there was an error invoking the script (for example, script not found).

Post-processing points to consider

Consider the following points when performing post processing.

- There is no feedback from the post-processing script. The standard and error output streams of the process are silently discarded.
- The return code is not checked.
- Errors are not published and the document is not rejected if any problems occur within the script.
- Post processing is not reliable. The script is not retried in case of failure.

Manage file system integration

Back-end file systems are a popular integration method for the trading engine to pick up messages to send partners and deposit messages received from partners. The following topics describe some file system integration management methods.

- Set up default file system integration
- Use file system to set meta-data values on page 230
- Post-processing to route inbound messages on page 230

Set up default file system integration

Use this procedure to create default file system integration exchanges for all document types (EDI, XML and binary). The structure of the directories for these exchanges is similar to the default file system integration for Cyclone Activator 4.2.x.

The default method creates three integration pickup exchanges and three integration delivery exchanges.

Steps

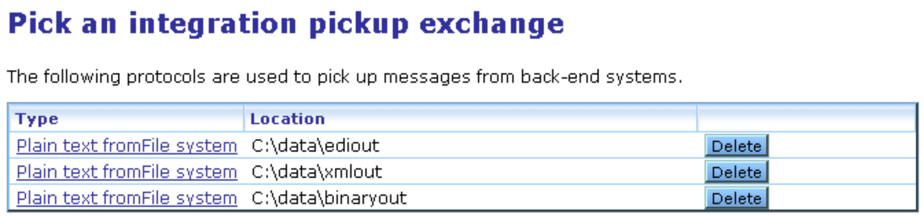
1. At the bottom of the community summary page, click **Add default integration exchanges** to open the Add default integration exchanges wizard.

2. Click **Next**.

3. Specify the top level directory path for the default directories (for example in Windows, c:\data). You can type the path or click the **Browse** button to select a directory. The trading engine creates the top level directory if it does not exist, as well as the document type-specific subdirectories.

4. Click **Finish** to create the integration exchanges and the file system directories.

   To check the integration exchanges, click **Integration delivery** or **Integration pickup** on the navigation graphic at the top of the community summary page. Figure 47 and Figure 48 show the default integration exchanges in the user interface.

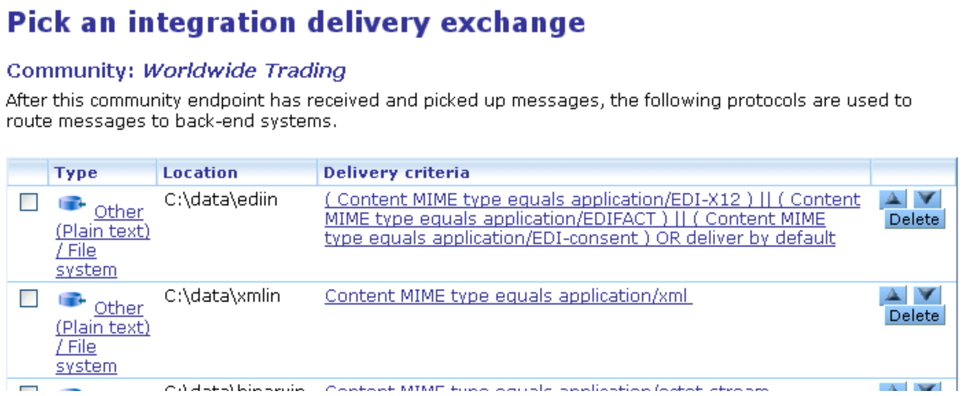


Pick an integration pickup exchange

The following protocols are used to pick up messages from back-end systems.

Type	Location	
Plain text fromFile system	C:\data\ediout	Delete
Plain text fromFile system	C:\data\xmlout	Delete
Plain text fromFile system	C:\data\binaryout	Delete

Figure 47. Default integration pickup exchanges



Pick an integration delivery exchange

Community: *Worldwide Trading*

After this community endpoint has received and picked up messages, the following protocols are used to route messages to back-end systems.

	Type	Location	Delivery criteria	
☐	Other (Plain text) / File system	C:\data\ediin	(Content MIME type equals application/EDI-X12) || (Content MIME type equals application/EDIFACT) || (Content MIME type equals application/EDI-consent) OR deliver by default	Delete
☐	Other (Plain text) / File system	C:\data\xmlin	Content MIME type equals application/xml	Delete

Figure 48. Default integration delivery exchanges

You can check the file system for the directories that have been created. The following is an example of the default file system directory structure on Windows.

c:\	data\	binaryin
		binaryout
		ediin
		ediout
		xmlin
		xmlout

Directories with a suffix of “out” are where your back-end system writes messages for the trading engine to pick up and then package and send to partners.

Directories with a suffix of “in” are where the trading engine routes messages from partners based on the content MIME type of the inbound messages. For inbound binary messages, the trading engine takes the further step of creating binaryin subdirectories named for the routing IDs of the sender and receiver. The trading engine creates these directories upon parsing sender and receiver information in the headers of inbound binary messages. The following is an example of the binaryin structure on Windows. You do not have to create these subdirectories yourself.

```
c:\   data\   binaryin\   sender A ID\   receiver ID
                          sender B ID\   receiver ID
                          sender C ID\   receiver ID
```

5 Once the default integration exchanges have been created, there are additional setup tasks for some exchanges. If you do not plan on using these exchanges, skip this step.

binaryout exchange

Creating binaryout subdirectories named for the routing IDs of the sender and receiver is recommended. These subdirectories can be made to correspond to meta-data attributes that already have been set up for you. This makes it possible for the trading engine to determine the sender and receiver of binary messages.

In the file system, create binaryout subdirectories named for the routing IDs of your community and of each partner to whom you plan to send binary messages. The following is an example of the structure on Windows.

```
c:\   data\   binaryout\   sender ID\   receiver A ID
                                        receiver B ID
                                        receiver C ID
```

Your back-end system needs to write outbound messages to the lowest level binaryout directory for a specific community and partner pair. For example:

c:\data\binaryout\senderID\receiver A ID

The trading engine, upon picking up a message, uses the names of the sender and receiver routing ID subdirectories as values for sender and receiver meta-data attributes. How this works is explained in Directory mapping on page 294

xmlout exchange

Setting up sender and receiver XPaths is recommended. This enables the trading engine to locate the sender and receiver information in outbound XML messages. This is not applicable for ebXML messages.

Click **Integration pickup** on the navigation graphic at the top of the community summary page. Then click the xmlout file system exchange to open the maintenance page. Select the from address tab.

Select **Always parse for the address** and **If the document is XML, use XPaths to locate the addresses**. Select a document type or click **XPath wizard** and point the wizard to the sender address in a sample XML document. Click **Save XPaths**.

Select the to address tab and do the same procedure to select the receiver XPath.

Use file system to set meta-data values

You can use a directory hierarchy in your file system that dictates values of meta-data elements associated with messages picked up through file system exchanges. For instance, the directory hierarchy can specify a document's sender, receiver or values for any number of other meta-data elements.

For configuration details see Directory mapping on page 294.

Post-processing to route inbound messages

You can use a post-processing script to route inbound binary messages and inbound messages based on MIME type. The following is a sample script for Windows that examines a message's content to determine where it should be moved after it is unpackaged. The script uses c:\dest as a sample root directory for receiving inbound messages; you should replace this with a location suitable for your installation. If you are running a multiple computer cluster, the destination should be a directory on a shared network drive.

In the sample script, the trading engine examines a message's MIME type. If the MIME type is known, the message is routed to an appropriate directory. If the MIME type is not known, the message is considered binary and routes to a directory based on the community routing ID and the partner routing ID.

You must manually create all target directories before using the post-processing script.

```
@echo off
rem Batch file to move file to directory based on received files
rem MIME type.

rem set the default destination directory to
rem c:\dest\binary\ReceiverRoutingId\SenderRoutingId.
rem Where ReceiverRoutingId is the Routing Id of the Receiving Community,
rem and SenderRoutingId is the Sender Routing Id of the Sending Party.

set destdir=c:\dest\binary\%2\%1

rem If the MIME Type of the content is known, then override the destination
rem directory.
if '%7' == 'application/EDI-X12' set destdir=c:\dest\x12
if '%7' == 'application/EDIFACT' set destdir=c:\dest\edifact
if '%7' == 'application/xml' set destdir=c:\dest\xml
if '%7' == 'text/xml' set destdir=c:\dest\xml

echo Moving %3\%4 to %destdir%
move %3\%4 %destdir%
```

Enable, disable, test exchanges

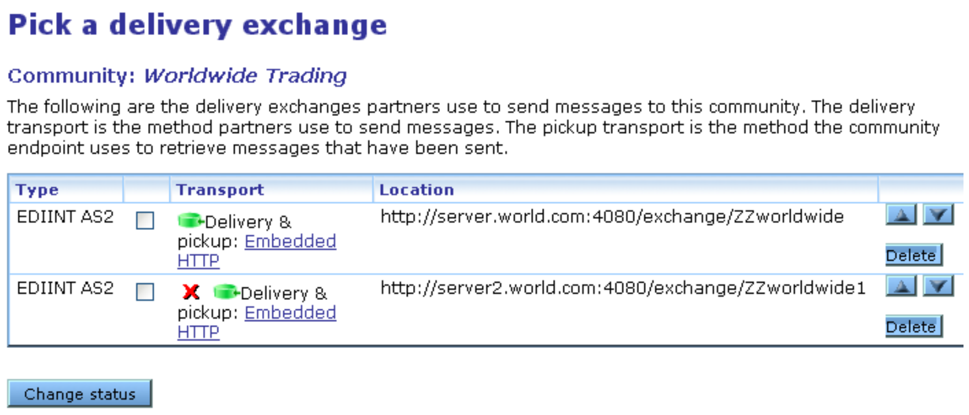
By default all delivery exchanges are enabled when you add them. To disable one, open the maintenance page for a delivery exchange, clear the check box for **Enable this delivery exchange** check box and click **Save changes**.

Another way to enable or disable exchanges is to open a maintenance page that lists configured exchanges and change their status. The following is how to open the maintenance pages that list exchanges.

Exchange	Opening maintenance page
Receive messages from partners	On the navigation graphic at the top of the community summary page, click **Delivery exchange**.
Integration pickup	On the navigation graphic at the top of the community summary page, click **Integration pickup**.

Exchange	Opening maintenance page
Integration delivery	On the navigation graphic at the top of the community summary page, click **Integration delivery**.
Send messages to partners	On the navigation graphic at the top of the partner summary page, click **Delivery exchange**.

Once a maintenance page is displayed, select a check box for a transport and click **Change status**. If active, this makes the transport inactive. If inactive, this makes the transport active.


Pick a delivery exchange

Community: *Worldwide Trading*

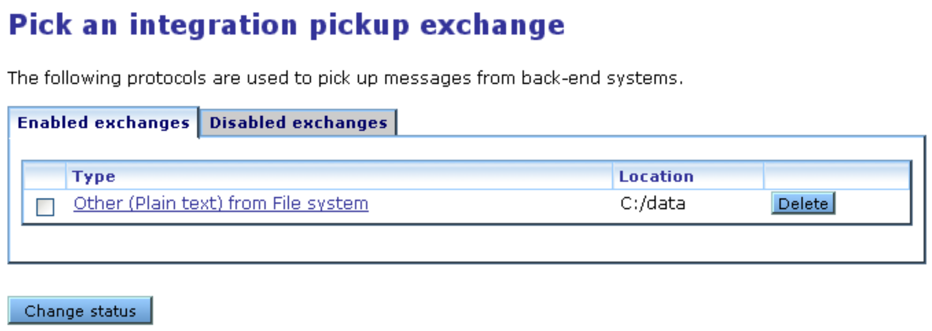
The following are the delivery exchanges partners use to send messages to this community. The delivery transport is the method partners use to send messages. The pickup transport is the method the community endpoint uses to retrieve messages that have been sent.

Type		Transport	Location	
EDIINT AS2	☐	Delivery & pickup: Embedded HTTP	http://server.world.com:4080/exchange/ZZworldwide	Delete
EDIINT AS2	☐	✗ Delivery & pickup: Embedded HTTP	http://server2.world.com:4080/exchange/ZZworldwide1	Delete

Figure 49. Community delivery exchange maintenance page

✗ A red X indicates a delivery exchange is disabled. If an X is not visible, the exchange is active.

For integration pickup exchanges, the user interface displays a maintenance page (Figure 50) with two tabs, one for enabled exchanges and the other for disabled exchanges. This is an especially helpful if you have many integration pickup exchanges, but only some are enabled at a given time.


Pick an integration pickup exchange

The following protocols are used to pick up messages from back-end systems.

	Type	Location	
☐	Other (Plain text) from File system	C:/data	Delete

Figure 50. Pick an integration pickup exchange page

At the bottom of community summary pages are links that let you enable or disable at the same time all pickup exchanges for receiving from partners.

When you open the maintenance page for a delivery exchange, the trading engine tests the connection to the exchange. Test results of success or failed display in the test results field. If the test fails, the trading engine displays an error message plus guidance for troubleshooting the connection problem.

For a test of an HTTP transport, only the connection is tested. The test does not validate a user name and password, if assigned.

If a test result is **not applicable**, it means the connection does not need testing.

Set delivery exchange preferences

On delivery exchange maintenance pages you can order transports according to preferences using the up and down arrows to the right of the exchange names.

On maintenance pages for partner delivery exchanges, the transport that is first in the list is the default the system uses for sending messages to the partner.

On maintenance pages for community trading delivery exchanges, the order of the transports as displayed is preserved when the community profile is exported. When a partner who also uses Cyclone Activator 5.0 or later imports the profile, the transport at the top of the list becomes the default for sending messages.

❖ ❖ ❖

14 Staged HTTP

Communities can used the staged HTTP web servlet transport to receive messages from partners. This transport is available only with the AS2 and ebXML message protocols.

Using this transport requires having a web server application running in your organization's DMZ and thorough operations knowledge of the web server.

Concepts

Procedure

Overview of staged HTTP

The staged HTTP transport provides a secure way to receive messages from partners without having to change firewall configurations. You can use this transport after deploying the staged HTTP servlet on a web server in your DMZ. You also must add staged HTTP as an inbound transport in your community profile.

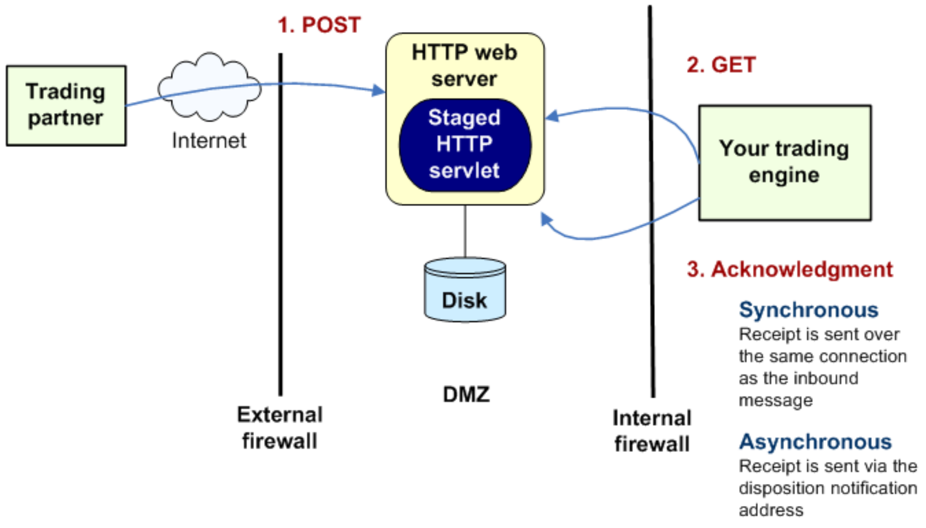


Figure 51. Using staged HTTP to receive messages from a partner

Figure 51 shows how this transport works, and the following describes the process.

1 Your partner sends a document to the web server using a standard message protocol (AS2). The staged HTTP servlet writes the file to disk pending retrieval by your trading engine. This provides failover protection.

2 At the usual polling interval, your trading engine polls the web server and retrieves the document with a GET method. This is the default manner for picking up documents. Alternately, you can configure the staged HTTP servlet to forward the incoming message to the trading engine immediately. This bypasses the polling interval, allowing faster throughput, but opens a port into your trusted network. (See File forwarding to bypass polling on page 243.)

3 Your trading engine sends a receipt to acknowledge receiving the document. The receipt can be sent synchronously over the same connection as the inbound message or asynchronously via the disposition notification address in the header of the inbound document.

The staged HTTP transport is only for receiving documents from partners. To send documents in a similar manner, contact technical support and ask about the optional Secure Webmailbox Server add-on.

Staged HTTP configuration outline

1. Deploy the staged HTTP servlet on your web server. See Staged HTTP files to deploy on page 237 and Deploy the web servlet on page 237.
2. Log on to the HTTP servlet user interface. See Log on to servlet UI on page 239.
3. Add a mailbox for your partner. See Add a mailbox on page 240.
4. For your community profile in the trading engine user interface, add a delivery exchange for receiving messages from the partner. See Staged HTTP fields on page 242.

Staged HTTP files to deploy

The files to deploy on the web server for the staged HTTP servlet are in [install directory]\[build number]\util \stagedhttp of the trading engine directory tree.

WebLogic users should use the TAR file in the 1.3 or 1.4 folder, depending on whether your version of WebLogic supports JRE 1.3 or 1.4.

Deploy the web servlet

Use this procedure to deploy the staged HTTP servlet on a supported web server application. See Staged HTTP files to deploy to find the servlet files in the trading engine directory tree. The servlet supports the following web servers:

- WebLogic 7.0.2 or higher
- WebLogic 8.1 or higher

After deploying the servlet on the web server, do not edit any of the servlet files, except as recommended by the user documentation or technical support. In particular, do not change or move the activation.xml file. This is the license file for the servlet. Changing this file will make the servlet inoperable.

The servlet uses three file system directories for processing messages and logging information. Servlet files control the paths for these directories. The following are the default directory paths. The name of the file that defines each path also is listed. You can change the path by editing the file.

Directory type	Default path	File that defines path
Message processing	**Windows**: C:\opt\cyclone\data\stagedhttp **Unix**: /opt/cyclone/data/stagedhttp	conf\ configurationrules. xml
Message temporary	**Windows**: C:\opt\cyclone\tmp\mailbox-data **Unix**: /opt/cyclone/tmp/mailbox-data	conf\ webmailboxconfig. xml
Logging	**Windows**: C:\var\log\cyclone\webmailbox **Unix**: /var/log/cyclone/webmailbox	conf\ log4j.properties

The message directories are created when the servlet is deployed and running. You must create the log directory before deploying the servlet. On UNIX make sure the user has permissions to add directories.

Steps

1 Create a file system directory for the servlet log files. The default path is:

Windows:
C:\var\log\cyclone\webmailbox

Unix:
/var/log/cyclone/webmailbox

2 Have your web server administrator add the following users in the web server application.

mailboxadmin

This user can fully configure mailboxes and can change global configuration settings.

communityadmin

This user can add, change and delete mailboxes. Mailboxes are created using template rules that the servlet controls. Little deviation from the template is allowed.

ecengine

This is the user for the trading engine to get documents. This role accesses mailboxes to get messages and then delete the retrieved messages from the staged HTTP servlet.

3 Deploy the staged HTTP servlet on a WebLogic server. If it does not already exist, create the following home directory:

UNIX:
/opt/cyclone

Windows:
c:\opt\cyclone

4 Create a subdirectory of the home directory named **stagedhttp**.

5 Extract the contents of the TAR file and copy the files to the stagedhttp directory. See Staged HTTP files to deploy on page 237 for which archive file to use.

Extract the TAR file using the tar command on UNIX or using an application such as WinZip on Windows.

6 Restart the web server.

7 Deploy the staged HTTP servlet as a web application using the WebLogic console application. Make sure the servlet is deployed with the **nostage** staging mode.

8 Log on to the staged HTTP servlet user interface. See Log on to servlet UI on page 239.

Log on to servlet UI

Do the following to log on to the staged HTTP servlet user interface after deploying the servlet on the web server.

Point a web browser to the web server with a URL in the following format:

http://[web server host]:[port]/stagedhttp/config

Note: Used **stagedhttp** in the URL if you followed the deployment recommendation to use this as a directory name for the servlet on the web server. Otherwise, use the name you selected.

Type the user name and password of the mailbox administrator when prompted.

The browser displays the staged HTTP servlet page in Figure 52.

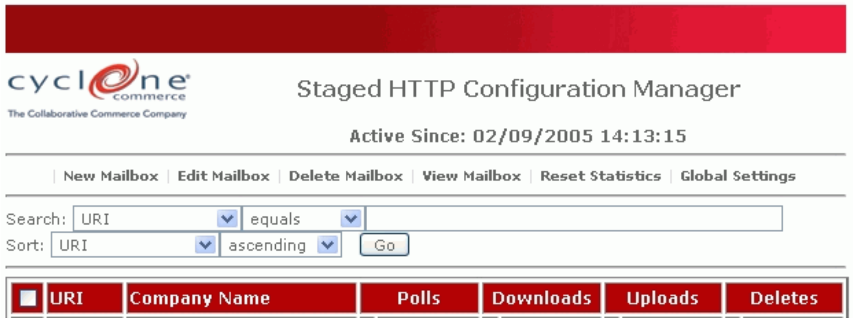

cyclone commerce
The Collaborative Commerce Company
Staged HTTP Configuration Manager
Active Since: 02/09/2005 14:13:15
New Mailbox | Edit Mailbox | Delete Mailbox | View Mailbox | Reset Statistics | Global Settings
Search: URI equals
Sort: URI ascending Go
URI
Company Name
Polls
Downloads
Uploads
Deletes

Figure 52. Staged HTTP servlet user interface

Managing mailboxes

You can use the staged HTTP servlet user interface to manage mailboxes and the files they contain. A mailbox is a web server directory. Partners connect to the web server via the URL for a specific mailbox. Files the partner sends are written to a temp directory and then moved to the mailbox when fully received. The sponsor's trading engine retrieves the files from the mailbox with the same URL.

The UI lets you:

- Add, edit and delete mailboxes
- View files in mailboxes
- View running totals for file uploads, downloads and deletions.
- View and change global settings for the servlet

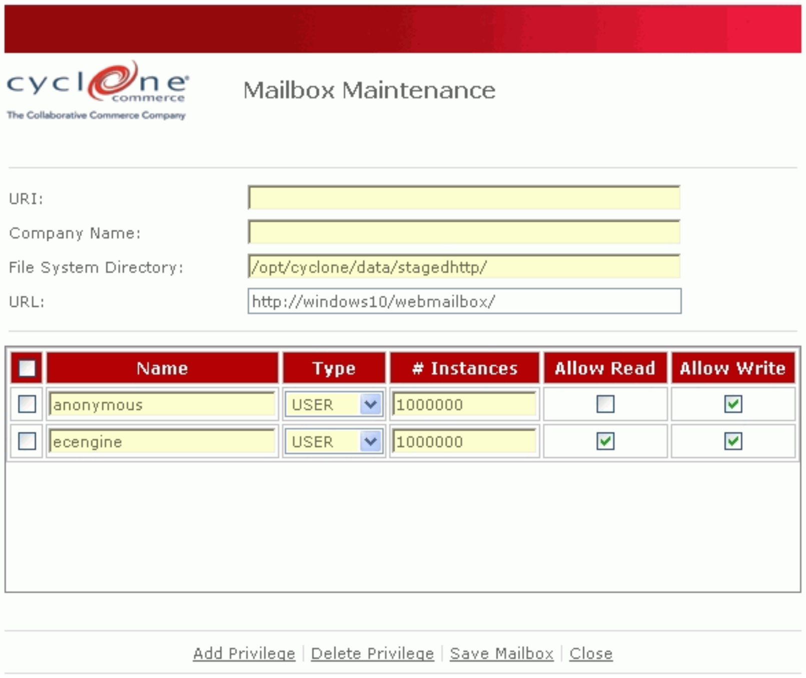
The following topics describe how to perform various tasks.

Add a mailbox

Use this procedure to add a mailbox.

Steps

1 Click **New Mailbox** at the top of the staged HTTP configuration manager page to display the mailbox maintenance page.



URI:

Company Name:

File System Directory: /opt/cyclone/data/stagedhttp/

URL: http://windows10/webmailbox/

	Name	Type	# Instances	Allow Read	Allow Write
☐	anonymous	USER	1000000	☐	✓
☐	ecengine	USER	1000000	✓	✓

Figure 53. Mailbox maintenance page

2 In the URI field type a directory name where inbound files will be written. As you type the name, it is appended to the URL in the URL field.

The partner connects with this URL to POST documents to the servlet directory. The trading engine connects with the same URL to pick up documents.

When you give the URL to the partner or the trading engine administrator, you must add the domain and port of the web server. For example, the mailbox maintenance page displays a URL like this:

http://webserver/webmailbox/partner

But you must add the domain and port to the URL before passing it along. The URL you provide must be in this format:

http://webserver**.domain.com:port**/webmailbox/partner

3 In the Company Name field type the name of the partner who will use this mailbox.

4 Select the users who can access this mailbox with their user IDs and passwords. The **anonymous** and **ecengine** users are displayed by default, but not selected. To add a user not displayed, click **Add Privilege**. You can select to allow users read and write access.

The **# instances** permission for users represents the number of clients that can concurrently access the mailbox. In most cases, make this a large number (for example, 1000000) to make sure partners can connect to the servlet.

5 Click **Save Mailbox** to close the mailbox maintenance page and save the mailbox.

Edit, delete, view mailboxes

To edit, delete or view contents of mailboxes, select a mailbox on the staged HTTP configuration manager page and click the action you want on the top toolbar.

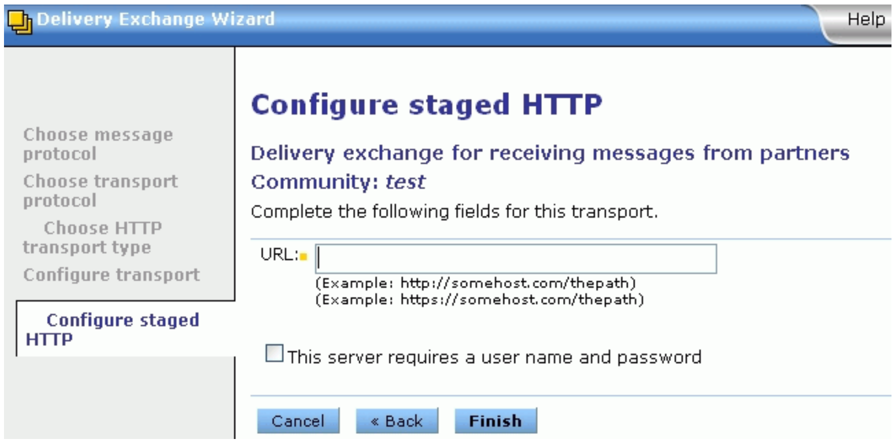
Global settings

There are no settings requiring attention on the global settings page, with the possible exception of the synchronous request timeout setting. This controls how long a connection stays open while waiting for a response over the same connection. An advanced user may have a reason for changing the default time of 600 seconds.

Staged HTTP fields

The following fields are used in the delivery exchange wizard in the trading engine user interface for configuring this transport.

The URL to use is the one set up in the staged HTTP servlet user interface for the mailbox the partner has for sending documents. The user ID and password to use are those authorized in the servlet UI for the mailbox.



Configure staged HTTP

Delivery exchange for receiving messages from partners

Community: *test*

Complete the following fields for this transport.

URL:

(Example: http://somehost.com/thepath)
(Example: https://somehost.com/thepath)

☐ This server requires a user name and password

Figure 54. Add staged HTTP transport

URL

The URL for connecting to the server.

This server requires a user name and password

If selected, type a user name and password for the community to connect to the server.

File forwarding to bypass polling

The default way to receive messages via the staged HTTP transport is when the trading engine polls the web server for inbound messages sent by partners. You can set up another way, where the web server forwards messages to the trading engine upon receipt. This bypasses polling and increases throughput, but opens a port into your trusted network.

To use forwarding, set up a staged HTTP transport for receiving messages in the community profile as though you are using the polling method. But also add an embedded HTTP server transport in the profile. The staged HTTP servlet forwards messages to the embedded server, but polling remains active with the staged HTTP transport. This provides failover protection. If a message cannot be forwarded because, for example, the trading engine server is not running, the message will be retrieved once the server restarts and polls the web server.

To enable forwarding, configure the forwarding.xml file in the staged HTTP servlet's conf directory. The following describes the elements in the file. You must delete or comment out unused elements.

HTTP Info

The information for forwarding to an embedded HTTP server is placed within this element. If the trading engine server runs on a multiple computer cluster, one HTTP Info block is needed for each computer. See the URL element.

uri="/SOMEMAILBOX"

The name of the mailbox created in the staged HTTP servlet user interface.

class="com.cyclonecommerce.webmailbox.forwarding.HttpForwardingHandler">

The Java class for the forwarding function.

<URL>http://node1:4080/exchange</URL>

This is the URL for connecting to the embedded HTTP server in the trading engine. The URL must be in the following format:

http://[host]/:[port]/exchange/[community routing ID]

You can copy most of the URL by looking up the embedded HTTP settings tab on the transport maintenance page in the trading engine user interface. The UI uses <cluster machines> for [host] in the URL in lieu of a machine name because the trading engine server may run on multiple computers. You must substitute the computer name. If the trading engine server runs on multiple computers, use an HTTP Info element with a different host in the URL for each machine. An example of this set up is shown in the forwarding.xml file.

<UserName>SOMEUSER</UserName>

If required, the user name for connecting to the embedded HTTP server.

<Password>SOMEPASSWORD</Password>

If required, the password for connecting to the embedded HTTP server.

<ProxyHost>MYPROXYHOST</ProxyHost>

If the staged HTTP server must connect through a proxy to the embedded server, the proxy name.

<ProxyPort>8080</ProxyPort>

If the staged HTTP server must connect through a proxy to the embedded server, the proxy port.

<ProxyUserName>SOMEPROXYUSER</ProxyUserName>

If the staged HTTP server must connect through a proxy to the embedded server, the user name to connect to the proxy.

<ProxyPassword>SOMEPROXYPASSWORD</ProxyPassword>

If the staged HTTP server must connect through a proxy to the embedded server, the password to connect to the proxy.

<ResponseTimeout>600000</ResponseTimeout>

The time in milliseconds the sender waits for a response before dropping the connection.

High availability staged HTTP

Figure 55 shows the staged HTTP servlet in a high availability configuration to handle inbound traffic. This involves setting up servlets on multiple web servers. A load balancer is used to direct traffic to each web server. The partner uses the virtual IP address of the load balancer in sending messages.

In this clustered example, each instance of the trading engine needs a staged HTTP delivery exchange for receiving messages from partners.

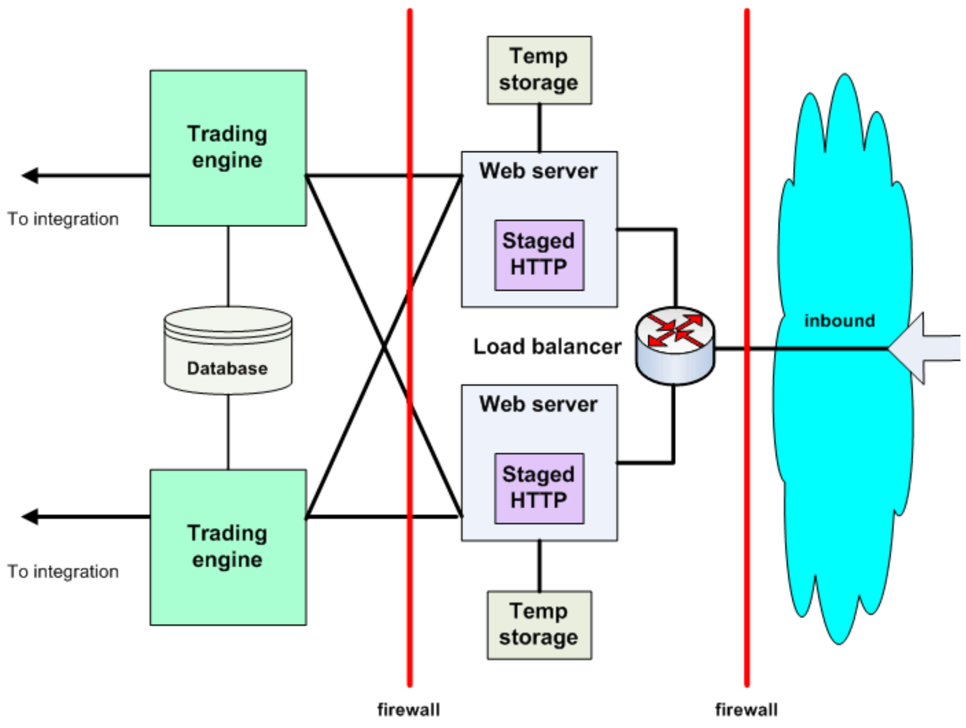


Figure 55. Staged HTTP high availability configuration

The following describes the process illustrated in the figure.

1 Inbound documents are routed to either staged HTTP servlet.

2 The staged HTTP servlet receives the request and persists it to temporary storage (non-clustered storage). If the message requires a synchronous acknowledgement, the inbound connection is held open.

3 Each instance of the trading engine is configured to poll each staged HTTP servlet.

4 The trading engine receives inbound document by issuing:

a. HTTP GET for listing of all inbound documents.

b. HTTP GET for each inbound document

c. HTTP DELETE to remove the document from temporary storage.

5 If the received message requires a synchronous acknowledgement, the trading engine produces the acknowledgement back to the consuming staged HTTP servlet so it can be produced back to the original inbound connection.

❖ ❖ ❖

15 Transport maintenance

After using the delivery exchange wizard to add a transport, you can use maintenance pages to change, test, activate, deactivate or delete community or partner exchanges. The wizard and maintenance pages have many of the same configuration fields. Generally, however, the wizard offers only the most commonly used fields for a transport. The maintenance pages have all configuration fields that can be changed. After using the wizard to add a transport, you might have to go the transport's maintenance page to fine tune the configuration.

Concepts

- Opening maintenance pages on page 248

Procedure

- Change embedded transports on page 302

Pages and fields

- SMTP maintenance on page 248
- HTTP maintenance on page 255
- Staged HTTP maintenance on page 260
- FTP maintenance on page 262
- SFTP maintenance on page 270
- File system maintenance on page 275
- JMS maintenance on page 279
- MQSeries maintenance on page 284
- Web services API maintenance
- Proxy tab on page 290
- From address and To address tabs on page 291
- Message attributes tab on page 293
- EDI Splitter tab on page 299
- Inline processing tab on page 300
- Delivery criteria tab on page 300

Opening maintenance pages

The following explains how to open delivery exchange maintenance pages.

Open maintenance page for trading exchange

To open the maintenance page for a trading delivery exchange, on a community or partner summary page, click **Delivery exchange** on the navigation graphic at the top of the page. Then click the exchange you want to open the maintenance page.

Open maintenance page for integration delivery

To open the maintenance page for a delivery integration exchange, on a community summary page, click **Integration delivery** on the navigation graphic at the top of the page. Then click the exchange you want to open the maintenance page.

Open maintenance page for integration pickup

To open the maintenance page for a pickup integration exchange, on a community summary page, click **Integration pickup** on the navigation graphic at the top of the page. Then click the exchange you want to open the maintenance page.

SMTP maintenance

The following topics document the fields on the maintenance pages for SMTP transports.

The maintenance pages display the transport fields that can be changed, while the delivery exchange wizard has only the most commonly used. The following are the fields on the settings and advanced tabs. For information about fields on the other maintenance tabs, see the topics for the specific tabs.

Embedded SMTP settings tab

Embedded SMTP server

A link is provided to view the settings for a particular embedded server or for the global embedded server.

Host

The fully qualified domain name of the computer on which the embedded SMTP server runs. This field cannot be changed.

Host name used by partners

The e-mail address that partners should use to access your embedded SMTP server. The trading engine maps this e-mail address to the correct delivery exchange.

While this field is visible on the delivery exchange maintenance page, you can only edit it on the embedded server maintenance page. See Change embedded transports on page 302.

E-mail address

The e-mail address that the remote partner uses to send messages to your local community.

Advanced tab (Embedded SMTP settings)

Back up the files that go through this transport

Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

Maximum files per polling interval

The highest number of messages the system can retrieve each time it polls.

Polling interval (seconds)

The interval in seconds the system waits before polling for messages to retrieve.

SMTP settings tab (community)

E-mail address

The e-mail address that the remote partner uses to send messages to your local community.

SMTP settings tab (partner)

E-mail address

The e-mail address for sending messages to a partner.

Use the global external SMTP server

Selecting this means the system’s external SMTP server is used to send messages to the partner. The link to configure the system’s external SMTP server is on the system management page of the user interface.

See Configuring external SMTP server on page 24.

Use a partner-specific SMTP server

Selecting this means you can specify an external SMTP server to send messages to the partner that is different from the system’s external SMTP server.

SMTP server

Enter an SMTP server for sending messages just for to this partner. You must type a fully qualified domain name or IP address for the server. If you leave this field blank, the system inserts its external SMTP server.

Port

The default port for sending mail is 25.

This server requires a user name and password

Select this if a user name and password are required to connect to the server and then complete the following fields. SMTP servers usually do not require user names and passwords for sending.

User name

The user name to connect to the server.

Password

The password to connect to the server.

Confirm password

The password to connect to the server.

Advanced tab (partner)

Maximum messages per connection

This is a way to allocate messages among nodes in a clustered environment. When the maximum is reached, another node is contacted to take messages and so on. This field is not applicable in a single node environment.

Retries

This is the number of times the trading engine will retry connecting to the partner's transport if the initial attempt to connect and send the message failed. The following are common reasons for triggering retries.

1. The connection attempt failed immediately for a reason such as host not found.
2. The host was found, but the connection process took longer than the connect timeout interval specified on the Advanced tab.
3. The connection was successful, but the partner's HTTP server took longer than the response timeout interval to return a 200 OK response indicating the message was successfully received. A 200 OK response is a transport response, separate from a message protocol response such as an AS2 receipt.

Note that in the last case, the 200 OK response also will include the receipt if synchronous receipts were requested. Otherwise, it will be a simple 200 OK response with no payload. And if an asynchronous receipt was requested, the partner will connect later to send it.

Retries occur according to an algorithm that starts at 5 minutes. The interval between retries increases with each subsequent retry in this pattern: 10 minutes, 15 minutes, 30 minutes, 60 minutes. The interval plateaus at 60 minutes. This means if the retry value is greater than 5, the fifth and each subsequent retry occurs at 60 minute intervals.

For example, if retries is 3, the system will try connecting again in 5 minutes if the initial connection attempt fails. If this retry attempt also fails, the system attempts a second retry in 10 minutes. The third retry attempt is made 15 minutes later. If the third retry attempt fails, the message is given a failed status. So

after four attempts (the first attempt plus 3 retries), the message fails. You can search for and manually resubmit failed messages in Message tracker.

Retries do not occur precisely at these intervals because each connection attempt takes some seconds, which varies by computer. So retries actually occur after the connection attempt time plus the interval.

This control applies only to retrying to send messages, not receiving. It applies only to retrying to send related to transport issues. It does not apply to successfully sent messages for which receipts have not been received as expected. Another control, resends, determines how many times the system will resend a message when a receipt is not received from the partner. For information about resends, see reliable messaging in the collaboration settings chapter.

Connect timeout (seconds)

How long in seconds that the trading engine waits for a connection to the delivery exchange before the connection times out.

Read timeout (seconds)

How long in seconds that the trading engine waits to read data from the delivery exchange before terminating the connection.

Back up the files that go through this transport

Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

Post-processing script

The full path to an executable file that contains post-processing commands. This field is available for community delivery integration exchanges and partner trading delivery exchanges.

POP settings tab

POP server

The name of the POP server that the trading engine polls for messages sent by your partner. This must be a fully qualified domain name or IP address.

Port

The POP server port by default is 110.

User name

The user name to connect to the server.

Password

The password to connect to the server.

Confirm password

The password to connect to the server.

Advanced tab (POP settings)

Maximum messages per connection

This is a way to allocate messages among nodes in a clustered environment. When the maximum is reached, another node is contacted to take messages and so on. This field is not applicable in a single node environment.

Retries

This is the number of times the trading engine will retry connecting to the partner's transport if the initial attempt to connect and send the message failed. The following are common reasons for triggering retries.

1. The connection attempt failed immediately for a reason such as host not found.
2. The host was found, but the connection process took longer than the connect timeout interval specified on the Advanced tab.

3. The connection was successful, but the partner's HTTP server took longer than the response timeout interval to return a 200 OK response indicating the message was successfully received. A 200 OK response is a transport response, separate from a message protocol response such as an AS2 receipt.

Note that in the last case, the 200 OK response also will include the receipt if synchronous receipts were requested. Otherwise, it will be a simple 200 OK response with no payload. And if an asynchronous receipt was requested, the partner will connect later to send it.

Retries occur according to an algorithm that starts at 5 minutes. The interval between retries increases with each subsequent retry in this pattern: 10 minutes, 15 minutes, 30 minutes, 60 minutes. The interval plateaus at 60 minutes. This means if the retry value is greater than 5, the fifth and each subsequent retry occurs at 60 minute intervals.

For example, if retries is 3, the system will try connecting again in 5 minutes if the initial connection attempt fails. If this retry attempt also fails, the system attempts a second retry in 10 minutes. The third retry attempt is made 15 minutes later. If the third retry attempt fails, the message is given a failed status. So after four attempts (the first attempt plus 3 retries), the message fails. You can search for and manually resubmit failed messages in Message tracker.

Retries do not occur precisely at these intervals because each connection attempt takes some seconds, which varies by computer. So retries actually occur after the connection attempt time plus the interval.

This control applies only to retrying to send messages, not receiving. It applies only to retrying to send related to transport issues. It does not apply to successfully sent messages for which receipts have not been received as expected. Another control, resends, determines how many times the system will resend a message when a receipt is not received from the partner. For information about resends, see reliable messaging in the collaboration settings chapter.

Connect timeout (seconds)

How long in seconds that the trading engine waits for a connection to the delivery exchange before the connection times out.

Read timeout (seconds)

How long in seconds that the trading engine waits to read data from the delivery exchange before terminating the connection.

Back up the files that go through this transport

Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

Maximum files per polling interval

The highest number of messages the system can retrieve each time it polls.

Polling interval (seconds)

The interval in seconds the system waits before polling for messages to retrieve.

HTTP maintenance

The following topics document the fields on the maintenance pages for HTTP transports.

The maintenance pages display the transport fields that can be changed, while the delivery exchange wizard has only the most commonly used. The following are the fields on the settings and advanced tabs. For information about fields on the other maintenance tabs, see the topics for the specific tabs.

If you require information about SSL authentication, see SSL authentication on page 316.

Embedded HTTP settings tab

Embedded HTTP server

A link is provided to view the settings for a particular embedded server or for the global embedded server. If a particular server, you can change servers.

Local URL

This URL describes the local port and path the embedded HTTP server uses. A server starts on each computer in the cluster using this information. If you have only one computer, only one server is started.

URL used by partners

This URL is the one your partners use to send messages to this delivery exchange. It becomes part of your partner profile when you export it. The host, port and path may be different than the values in the local URL. If your network uses a load balancer or firewall, contact your network administrator for the correct value.

This URL should include the fully qualified host name or IP address, port number, and location where partners must connect to send documents.

Embedded HTTPS settings tab

Embedded HTTPS server

A link is provided to view the settings for the embedded server. You also can change servers.

Local URL

This URL describes the local port and path the embedded HTTP server uses. A server starts on each computer in the cluster using this information. If you have only one computer, only one server is started.

URL used by partners

This URL is the one your partners use to send messages to this delivery exchange. It becomes part of your partner profile when you export it. The host, port and path may be different than the values in the local URL. If your network uses a load balancer or firewall, contact your network administrator for the correct value.

This URL should include the fully qualified host name or IP address, port number, and location where partners must connect to send documents.

Advanced tab (community)

Back up the files that go through this transport

Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

HTTP or HTTPS settings tab (partner)

URL

The URL for connecting to the server.

Clients must use SSL to connect to this server

Select this to have Secure Sockets Layer protocol in use during connections. The server presents a certificate for verification. To do this, a certificate in a profile must be designated as the SSL certificate. The server must support SSL. If this is not selected, connections are anonymous.

Enable host name verification

If selected, the trading engine compares the name of the SSL server to the name in the server's certificate to ensure they are the same.

This server requires a user name and password

If selected, type a user name and password for the community to connect to the server.

Advanced tab (partner)

Maximum concurrent connections

If the system uses multiple JVM nodes, the maximum number of connections the system can make to this exchange at the same time. If you change this field, the number should at least equal the number of nodes.

Retries

This is the number of times the trading engine will retry connecting to the partner's transport if the initial attempt to connect and send the message failed. The following are common reasons for triggering retries.

1. The connection attempt failed immediately for a reason such as host not found.
2. The host was found, but the connection process took longer than the connect timeout interval specified on the Advanced tab.
3. The connection was successful, but the partner's HTTP server took longer than the response timeout interval to return a 200 OK response indicating the message was successfully received. A 200 OK response is a transport response, separate from a message protocol response such as an AS2 receipt.

Note that in the last case, the 200 OK response also will include the receipt if synchronous receipts were requested. Otherwise, it will be a simple 200 OK response with no payload. And if an asynchronous receipt was requested, the partner will connect later to send it.

Retries occur according to an algorithm that starts at 5 minutes. The interval between retries increases with each subsequent retry in this pattern: 10 minutes, 15 minutes, 30 minutes, 60 minutes. The interval plateaus at 60 minutes. This means if the retry value is greater than 5, the fifth and each subsequent retry occurs at 60 minute intervals.

For example, if retries is 3, the system will try connecting again in 5 minutes if the initial connection attempt fails. If this retry attempt also fails, the system attempts a second retry in 10 minutes. The third retry attempt is made 15 minutes later. If the third retry attempt fails, the message is given a failed status. So

after four attempts (the first attempt plus 3 retries), the message fails. You can search for and manually resubmit failed messages in Message tracker.

Retries do not occur precisely at these intervals because each connection attempt takes some seconds, which varies by computer. So retries actually occur after the connection attempt time plus the interval.

This control applies only to retrying to send messages, not receiving. It applies only to retrying to send related to transport issues. It does not apply to successfully sent messages for which receipts have not been received as expected. Another control, resends, determines how many times the system will resend a message when a receipt is not received from the partner. For information about resends, see reliable messaging in the collaboration settings chapter.

Connect timeout (seconds)

How long in seconds that the trading engine waits for a connection to the delivery exchange before the connection times out.

Read timeout (seconds)

How long in seconds that the trading engine waits to read data from the delivery exchange before terminating the connection.

Response timeout (seconds)

How long in seconds that the trading engine waits for the delivery exchange to respond to a request before terminating the connection.

Back up the files that go through this transport

Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

Post-processing script

The full path to an executable file that contains post-processing commands. This field is available for community delivery integration exchanges and partner trading delivery exchanges.

Staged HTTP maintenance

The following topics document the fields on the maintenance pages for the staged HTTP web servlet transport.

The maintenance pages display the transport fields that can be changed, while the delivery exchange wizard has only the most commonly used. The following are the fields on the settings and advanced tabs. For information about fields on the other maintenance tabs, see the topics for the specific tabs.

HTTP settings tab

URL

The URL for connecting to the server.

This server requires a user name and password

If selected, type a user name and password for the community to connect to the server.

Advanced tab

Maximum concurrent connections

If the system uses multiple JVM nodes, the maximum number of connections the system can make to this exchange at the same time. If you change this field, the number should at least equal the number of nodes.

Retries

This is the number of times the trading engine will retry connecting to the partner's transport if the initial attempt to connect and send the message failed. The following are common reasons for triggering retries.

1. The connection attempt failed immediately for a reason such as host not found.

2. The host was found, but the connection process took longer than the connect timeout interval specified on the Advanced tab.
3. The connection was successful, but the partner's HTTP server took longer than the response timeout interval to return a 200 OK response indicating the message was successfully received. A 200 OK response is a transport response, separate from a message protocol response such as an AS2 receipt.

Note that in the last case, the 200 OK response also will include the receipt if synchronous receipts were requested. Otherwise, it will be a simple 200 OK response with no payload. And if an asynchronous receipt was requested, the partner will connect later to send it.

Retries occur according to an algorithm that starts at 5 minutes. The interval between retries increases with each subsequent retry in this pattern: 10 minutes, 15 minutes, 30 minutes, 60 minutes. The interval plateaus at 60 minutes. This means if the retry value is greater than 5, the fifth and each subsequent retry occurs at 60 minute intervals.

For example, if retries is 3, the system will try connecting again in 5 minutes if the initial connection attempt fails. If this retry attempt also fails, the system attempts a second retry in 10 minutes. The third retry attempt is made 15 minutes later. If the third retry attempt fails, the message is given a failed status. So after four attempts (the first attempt plus 3 retries), the message fails. You can search for and manually resubmit failed messages in Message tracker.

Retries do not occur precisely at these intervals because each connection attempt takes some seconds, which varies by computer. So retries actually occur after the connection attempt time plus the interval.

This control applies only to retrying to send messages, not receiving. It applies only to retrying to send related to transport issues. It does not apply to successfully sent messages for which receipts have not been received as expected. Another control, resends, determines how many times the system will resend a message when a receipt is not received from the partner. For information about resends, see reliable messaging in the collaboration settings chapter.

Connect timeout (seconds)

How long in seconds that the trading engine waits for a connection to the delivery exchange before the connection times out.

Read timeout (seconds)

How long in seconds that the trading engine waits to read data from the delivery exchange before terminating the connection.

Response timeout (seconds)

How long in seconds that the trading engine waits for the delivery exchange to respond to a request before terminating the connection.

Back up the files that go through this transport

Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

Maximum files per polling interval

The highest number of messages the system can retrieve each time it polls.

Polling interval (seconds)

The interval in seconds the system waits before polling for messages to retrieve.

FTP maintenance

The following topics document the fields on the maintenance pages for the FTP transport.

The maintenance pages display the transport fields that can be changed, while the delivery exchange wizard has only the most commonly used. The following are the fields on the settings and advanced tabs. For information about fields on the other maintenance tabs, see the topics for the specific tabs.

If you require information about SSL authentication, see SSL authentication on page 316.

FTP or FTPS settings tab

FTP Server

The name of the FTP server.

Port

The port on which the server listens for incoming connections. The default is 21.

Pickup directory

Type the path of the directory on your server where messages are picked up. When the trading engine polls the server for files, it only looks in the pickup directory, not an inbox directory.

Use temporary files to avoid read/write collisions

We recommend using this option to prevent the trading engine from attempting to retrieve partially written files. When this is selected, you must select one of the two following options.

There may be some specialized servers, typically running on mainframes, that support only part of the FTP protocol (RFC 959). In such cases you may have to clear this check box and take steps of your own to make sure collisions do not occur.

Use separate directory for temporary files

Type the full path of an inbox directory (for example, c:\data\inbox). Files are uploaded to this directory. When fully written, files are moved to the pickup directory for retrieval.

Do not put the inbox under the pickup directory unless you use a period at the beginning of the inbox name. The trading engine and other applications ignore directories and files that begin with periods.

For example, do not use the following directory structure:

c:\data\pickup\inbox

But you can use the following because a period is the first character of the inbox directory name:

c:\data\pickup\.inbox

When receiving files from a partner, we recommend that your partner write files to the inbox directory first and then move them to the pickup directory when they are ready to be retrieved. This process is automatic if your partner also uses Cyclone Interchange or Activator. If the partner uses other software to upload files to your server, the software should be configured to initially upload the files to the inbox directory and move them to the pickup directory when they are ready to be retrieved.

For outbound integration, the back-end system must write the message to the inbox and then move it to the pickup directory.

For inbound integration and sending outbound to partners, the trading engine writes to the inbox and then moves the message to the pickup directory.

Use special extension in pickup directory for temporary files

If you prefer not to use an inbox, select this option. While a file is being written to the pickup directory, a temporary extension is added so the system knows not to retrieve it because the file is only partially written. Once fully written, the temporary extension goes away and the file can be retrieved.

User name

The user name to connect to the server.

Password

The password to connect to the server.

Confirm password

The password to connect to the server.

Select the file type

Specifies the format of the files that are transmitted over this delivery exchange. Available options are:

- Binary (default)
- ASCII - automatic line ending

- ASCII - user CR/LF
- ASCII - use LF only

This field is available on delivery exchange maintenance pages only.

Use passive mode

If selected, indicates that files should be transmitted using passive mode. Clear this check box to indicate active mode.

This field is available on delivery exchange maintenance pages only.

Clients must use SSL to connect to this server

Select this to have Secure Sockets Layer protocol in use during connections. The server presents a certificate for verification. To do this, a certificate in a profile must be designated as the SSL certificate. The server must support SSL. If this is not selected, connections are anonymous.

Enable host name verification

If selected, the trading engine compares the name of the SSL server to the name in the server's certificate to ensure they are the same.

Preserve original filenames

Select this if you want original file names to be preserved when the trading engine delivers messages.

For binary messages, we recommend that you preserve original file names. Otherwise, the trading engine assigns a unique file name that does not readily identify the contents of the file. Preserving original file names also allows your back-end application to process binary messages based on their file names.

This field applies to community integration delivery exchanges and partner delivery exchanges only.

Overwrite duplicate filenames

An option when you choose to preserve original file names. If duplicate file names are detected, the trading engine overwrites the existing file.

Sequentially number duplicate filenames

An option when you choose to preserve original file names. If duplicate file names are detected, the trading engine appends a number to the new file.

Generate unique filenames

Select to have the system provide a unique file name instead of using the original name.

This field applies to community integration delivery exchanges and partner delivery exchanges only.

When selected, files are given arbitrary names. The names always have less than 30 characters and often have less than 20 characters.

Appended to the file name is a hex representation of a monotonically increasing file name counter that is maintained in the database and guaranteed to be unique across all nodes in a cluster. In addition, if the original file name had an extension, the same extension is appended to the unique name the system generates.

The following are examples of unique file names generated by the system, one with the original file extension and one without:

dabeed45_4cb.edi
z47e4120_4ce

Advanced tab

Maximum concurrent connections

If the system uses multiple JVM nodes, the maximum number of connections the system can make to this exchange at the same time. If you change this field, the number should at least equal the number of nodes.

Retries

This is the number of times the trading engine will retry connecting to the partner's transport if the initial attempt to connect and send the message failed. The following are common reasons for triggering retries.

1. The connection attempt failed immediately for a reason such as host not found.
2. The host was found, but the connection process took longer than the connect timeout interval specified on the Advanced tab.
3. The connection was successful, but the partner's HTTP server took longer than the response timeout interval to return a 200 OK response indicating the message was successfully received. A 200 OK response is a transport response, separate from a message protocol response such as an AS2 receipt.

Note that in the last case, the 200 OK response also will include the receipt if synchronous receipts were requested. Otherwise, it will be a simple 200 OK response with no payload. And if an asynchronous receipt was requested, the partner will connect later to send it.

Retries occur according to an algorithm that starts at 5 minutes. The interval between retries increases with each subsequent retry in this pattern: 10 minutes, 15 minutes, 30 minutes, 60 minutes. The interval plateaus at 60 minutes. This means if the retry value is greater than 5, the fifth and each subsequent retry occurs at 60 minute intervals.

For example, if retries is 3, the system will try connecting again in 5 minutes if the initial connection attempt fails. If this retry attempt also fails, the system attempts a second retry in 10 minutes. The third retry attempt is made 15 minutes later. If the third retry attempt fails, the message is given a failed status. So after four attempts (the first attempt plus 3 retries), the message fails. You can search for and manually resubmit failed messages in Message tracker.

Retries do not occur precisely at these intervals because each connection attempt takes some seconds, which varies by computer. So retries actually occur after the connection attempt time plus the interval.

This control applies only to retrying to send messages, not receiving. It applies only to retrying to send related to transport issues. It does not apply to successfully sent messages for which receipts have not been received as expected. Another control, resends, determines how many times the system will resend a message when a receipt is not received from the partner. For information about resends, see reliable messaging in the collaboration settings chapter.

Connect timeout (seconds)

How long in seconds that the trading engine waits for a connection to the delivery exchange before the connection times out.

Read timeout (seconds)

How long in seconds that the trading engine waits to read data from the delivery exchange before terminating the connection.

User commands

Type user commands such as SITE to be sent to the server after login. Commands must be entered in the exact case and format expected by the server. For example, most FTP clients allow “mkdir test”, but most servers will only accept “MKD test”. Consult RFC 959 for a list of standard FTP commands. Specific servers may support other commands. Refer to the server documentation for more information.

If any command fails, the remaining commands will not be executed, and production to the FTP server will fail. To avoid possible failures, preface any command with an “at” sign (@) to indicate that errors from that command should be ignored, for example, “@MKD test”. Preface any command with an asterisk to cause the entire line to be treated as a comment, for example, “*Create test directory”.

This field is available for the FTP transport only.

Command set file

The FTP command set file controls the commands sent to the FTP server for operations such as send, receive, delete. The default command set file is **ftpcommandset.xml**. This field lets you specify a different command set file. In most cases you can use the default value. Changing this is only for advanced FTP users with specialized needs.

The field value is the name of an entry in filereg.xml in [install directory]\[build number]\conf that points to another file in the conf directory. The following is the entry in filereg.xml for the default ftpcommandset.xml file:

<File name="ftpcommandset.xml" path="conf/ ftpcommandset.xml"/>

If you want to add a new command set file, create the file and add an entry for it in filereg.xml. For example, if your new command set is called myftpcommandset.xml, enter that name in the field and add the following entry to filereg.xml:

<File name="myftpcommandset.xml" path="conf/ myftpcommandset.xml"/>

Attempt restarts

Indicates whether the system resumes transferring large files at the point interrupted when a connection is lost before a transfer is completed. If you select this check box, the system resumes processing of files at least as large as specified in the restartable minimum bytes field. This checkpoint-restart feature is worthwhile only for large documents. If this option is not used, the system starts a file transfer over when processing is interrupted.

Restartable minimum bytes (MB)

If attempt restarts is selected, the minimum size of a file that triggers the system to continue the file transfer at the point interrupted before the connection was lost. The minimum size is in megabytes. The system only resumes transfers of files that meet this minimum. The system starts over the transfer of smaller files whose processing is interrupted.

Temporary file lifetime (hours)

If attempt restarts is selected, how long the system retains a file whose transfer has been interrupted while waiting for the connection to be restored. This temporary file enables the system to resume the transfer at the point interrupted.

Back up the files that go through this transport

Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

Post-processing script

The full path to an executable file that contains post-processing commands. This field is available for community delivery integration exchanges and partner trading delivery exchanges.

Maximum files per polling interval

The highest number of messages the system can retrieve each time it polls.

Polling interval (seconds)

The interval in seconds the system waits before polling for messages to retrieve.

SFTP maintenance

The following topics document the fields on the maintenance pages for the SFTP transport.

The maintenance pages display the transport fields that can be changed, while the delivery exchange wizard has only the most commonly used. The following are the fields on the settings and advanced tabs. For information about fields on the other maintenance tabs, see the topics for the specific tabs.

SFTP settings tab

SFTP Server

The name of the SFTP server.

Port

The port on which the server listens for incoming connections. The default is 22.

Pickup directory

Type the path of the directory on your server where messages are picked up. When the trading engine polls the server for files, it only looks in the pickup directory, not an inbox directory.

Use temporary files to avoid read/write collisions

We recommend using this option to prevent the trading engine from attempting to retrieve partially written files. When this is selected, you must select one of the two following options.

Use separate directory for temporary files

Type the full path of an inbox directory (for example, c:\data\inbox). Files are uploaded to this directory. When fully written, files are moved to the pickup directory for retrieval.

Do not put the inbox under the pickup directory unless you use a period at the beginning of the inbox name. The trading engine and other applications ignore directories and files that begin with periods.

For example, do not use the following directory structure:

c:\data\pickup\inbox

But you can use the following because a period is the first character of the inbox directory name:

c:\data\pickup\.inbox

When receiving files from a partner, we recommend that your partner write files to the inbox directory first and then move them to the pickup directory when they are ready to be retrieved. This process is automatic if your partner also uses Cyclone Interchange or Activator. If the partner uses other software to upload files to your server, the software should be configured to initially upload the files to the inbox directory and move them to the pickup directory when they are ready to be retrieved.

For outbound integration, the back-end system must write the message to the inbox and then move it to the pickup directory.

For inbound integration and sending outbound to partners, the trading engine writes to the inbox and then moves the message to the pickup directory.

Use special extension in pickup directory for temporary files

If you prefer not to use an inbox, select this option. While a file is being written to the pickup directory, a temporary extension is added so the system knows not to retrieve it because the file is only partially written. Once fully written, the temporary extension goes away and the file can be retrieved.

Current public key

The RSA or DSA public key for the SFTP server. The trading engine uses the key to authenticate the server.

New public key

The path to a new file that contains the RSA or DSA public key for the SFTP server. You must get this file from the server administrator. If the server is modified to use a new public-private key pair, the public key must be updated.

Preserve original filenames

Select this if you want original file names to be preserved when the trading engine delivers messages.

For binary messages, we recommend that you preserve original file names. Otherwise, the trading engine assigns a unique file name that does not readily identify the contents of the file. Preserving original file names also allows your back-end application to process binary messages based on their file names.

This field applies to community integration delivery exchanges and partner delivery exchanges only.

Overwrite duplicate filenames

An option when you choose to preserve original file names. If duplicate file names are detected, the trading engine overwrites the existing file.

Sequentially number duplicate filenames

An option when you choose to preserve original file names. If duplicate file names are detected, the trading engine appends a number to the new file.

Generate unique filenames

Select to have the system provide a unique file name instead of using the original name.

This field applies to community integration delivery exchanges and partner delivery exchanges only.

When selected, files are given arbitrary names. The names always have less than 30 characters and often have less than 20 characters.

Appended to the file name is a hex representation of a monotonically increasing file name counter that is maintained in the database and guaranteed to be unique across all nodes in a cluster. In addition, if the original file name had an extension, the same extension is appended to the unique name the system generates.

The following are examples of unique file names generated by the system, one with the original file extension and one without:

dabeed45_4cb.edi
z47e4120_4ce

Advanced tab

Maximum concurrent connections

If the system uses multiple JVM nodes, the maximum number of connections the system can make to this exchange at the same time. If you change this field, the number should at least equal the number of nodes.

Retries

This is the number of times the trading engine will retry connecting to the partner's transport if the initial attempt to connect and send the message failed. The following are common reasons for triggering retries.

1. The connection attempt failed immediately for a reason such as host not found.
2. The host was found, but the connection process took longer than the connect timeout interval specified on the Advanced tab.
3. The connection was successful, but the partner's HTTP server took longer than the response timeout interval to return a 200 OK response indicating the message was successfully received. A 200 OK response is a transport response, separate from a message protocol response such as an AS2 receipt.

Note that in the last case, the 200 OK response also will include the receipt if synchronous receipts were requested. Otherwise, it will be a simple 200 OK response with no payload. And if an asynchronous receipt was requested, the partner will connect later to send it.

Retries occur according to an algorithm that starts at 5 minutes. The interval between retries increases with each subsequent retry in this pattern: 10 minutes, 15 minutes, 30 minutes, 60 minutes. The interval plateaus at 60 minutes. This means if the retry value is greater than 5, the fifth and each subsequent retry occurs at 60 minute intervals.

For example, if retries is 3, the system will try connecting again in 5 minutes if the initial connection attempt fails. If this retry attempt also fails, the system attempts a second retry in 10 minutes. The third retry attempt is made 15 minutes later. If the third retry attempt fails, the message is given a failed status. So after four attempts (the first attempt plus 3 retries), the message fails. You can search for and manually resubmit failed messages in Message tracker.

Retries do not occur precisely at these intervals because each connection attempt takes some seconds, which varies by computer. So retries actually occur after the connection attempt time plus the interval.

This control applies only to retrying to send messages, not receiving. It applies only to retrying to send related to transport issues. It does not apply to successfully sent messages for which receipts have not been received as expected. Another control, resends, determines how many times the system will resend a message when a receipt is not received from the partner. For information about resends, see reliable messaging in the collaboration settings chapter.

Read timeout (seconds)

How long in seconds that the trading engine waits to read data from the delivery exchange before terminating the connection.

Back up the files that go through this transport

Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

Post-processing script

The full path to an executable file that contains post-processing commands. This field is available for community delivery integration exchanges and partner trading delivery exchanges.

Maximum files per polling interval

The highest number of messages the system can retrieve each time it polls.

Polling interval (seconds)

The interval in seconds the system waits before polling for messages to retrieve.

File system maintenance

The following topics document the fields on the maintenance pages for the file system transport.

The maintenance pages display the transport fields that can be changed, while the delivery exchange wizard has only the most commonly used. The following are the fields on the settings and advanced tabs. For information about fields on the other maintenance tabs, see the topics for the specific tabs.

File system settings tab

Directory name

Use the **Browse** button or type the full path for a unique directory where the trading engine picks up or routes messages, depending on whether the transport is used for sending or receiving. If the directory does not exist, the trading engine creates it for you.

Use a unique directory name. You may want to give the directory a name that indicates whether the transport is being used for inbound or outbound integration, receiving messages from partners or sending messages to partners. For example, for outbound integration you could name the pickup directory \data\out; for inbound integration \data\in.

Preserve original filenames

Select this if you want original file names to be preserved when the trading engine delivers messages.

For binary messages, we recommend that you preserve original file names. Otherwise, the trading engine assigns a unique file name that does not readily identify the contents of the file. Preserving original file names also allows your back-end application to process binary messages based on their file names.

This field applies to community integration delivery exchanges and partner delivery exchanges only.

Overwrite duplicate filenames

An option when you choose to preserve original file names. If duplicate file names are detected, the trading engine overwrites the existing file.

Sequentially number duplicate filenames

An option when you choose to preserve original file names. If duplicate file names are detected, the trading engine appends a number to the new file.

Generate unique filenames

Select to have the system provide a unique file name instead of using the original name.

This field applies to community integration delivery exchanges and partner delivery exchanges only.

When selected, files are given arbitrary names. The names always have less than 30 characters and often have less than 20 characters.

Appended to the file name is a hex representation of a monotonically increasing file name counter that is maintained in the database and guaranteed to be unique across all nodes in a cluster. In addition, if the original file name had an extension, the same extension is appended to the unique name the system generates.

The following are examples of unique file names generated by the system, one with the original file extension and one without:

dabeed45_4cb.edi
z47e4120_4ce

Advanced tab

Maximum concurrent connections

If the system uses multiple JVM nodes, the maximum number of connections the system can make to this exchange at the same time. If you change this field, the number should at least equal the number of nodes.

Retries

This is the number of times the trading engine will retry connecting to the partner’s transport if the initial attempt to connect and send the message failed. The following are common reasons for triggering retries.

1. The connection attempt failed immediately for a reason such as host not found.
2. The host was found, but the connection process took longer than the connect timeout interval specified on the Advanced tab.
3. The connection was successful, but the partner’s HTTP server took longer than the response timeout interval to return a 200 OK response indicating the message was successfully received. A 200 OK response is a transport response, separate from a message protocol response such as an AS2 receipt.

Note that in the last case, the 200 OK response also will include the receipt if synchronous receipts were requested. Otherwise, it will be a simple 200 OK response with no payload. And if an asynchronous receipt was requested, the partner will connect later to send it.

Retries occur according to an algorithm that starts at 5 minutes. The interval between retries increases with each subsequent retry in this pattern: 10 minutes, 15 minutes, 30 minutes, 60 minutes. The interval plateaus at 60 minutes. This means if the retry value is greater than 5, the fifth and each subsequent retry occurs at 60 minute intervals.

For example, if retries is 3, the system will try connecting again in 5 minutes if the initial connection attempt fails. If this retry attempt also fails, the system attempts a second retry in 10 minutes. The third retry attempt is made 15 minutes later. If the third retry attempt fails, the message is given a failed status. So

after four attempts (the first attempt plus 3 retries), the message fails. You can search for and manually resubmit failed messages in Message tracker.

Retries do not occur precisely at these intervals because each connection attempt takes some seconds, which varies by computer. So retries actually occur after the connection attempt time plus the interval.

This control applies only to retrying to send messages, not receiving. It applies only to retrying to send related to transport issues. It does not apply to successfully sent messages for which receipts have not been received as expected. Another control, resends, determines how many times the system will resend a message when a receipt is not received from the partner. For information about resends, see reliable messaging in the collaboration settings chapter.

Back up the files that go through this transport

Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

Maximum files per polling interval

The highest number of messages the system can retrieve each time it polls.

Polling interval (seconds)

The interval in seconds the system waits before polling for messages to retrieve.

Post-processing script

The full path to an executable file that contains post-processing commands. This field is available for community delivery integration exchanges and partner trading delivery exchanges.

JMS maintenance

The following topics document the fields on the maintenance pages for the JMS transport.

The maintenance pages display the transport fields that can be changed, while the delivery exchange wizard has only the most commonly used. The following are the fields on the settings and advanced tabs. For information about fields on the other maintenance tabs, see the topics for the specific tabs.

JMS (polled) settings or JMS (listener) settings tab

JMS queue

The name of the queue. Example: XMLQueue@router1

This queue requires a user name and password

Select this if the queue requires a user name and password. When selected, fields appear for entering the information.

User name

A user name for the JNDI provider. This value could be blank and is typically provided for in the JNDI URL. However, this depends on the JNDI provider and how it is configured.

Password

A password for the JNDI provider. This value could be blank and is typically provided in the JNDI URL. However, this depends on the JNDI provider and how it is configured.

Confirm password

Type the password again for confirmation.

Use JNDI

Select this if a Java Naming and Directory Interface (JNDI) provider. When selected the applicable fields display.

JNDI URL

The network URL used to obtain access to the JNDI service provider for your JMS service. Example: smqp://localhost:4001/timeout=10000

JNDI factory

The name for the JNDI service provider class. Example: com.swiftmq.jndi.InitialContextFactoryImpl

This provider requires a user name and password

Select this if JMS requires a user name and password. When selected, fields appear for entering the information.

User name

A user name for the JMS provider. This can be the same as your JNDI user name. However, this depends on your JMS provider and how it is configured.

Password

A password for the JMS provider. This can be the same as your JNDI password. However, this depends on your JMS provider and how it is configured.

Confirm password

The password again for confirmation.

JMS connection factory

The connection factory as defined within the JMS provider. This value can be either in the form **factoryname@routername** or the JNDI public symbol for the QueueConnectionFactory. Examples: plainsocket@router1 or QueueConnectionFactory22. This depends on your JMS provider and how it is configured.

Use a custom Java implementation

Select this for a JMS provider that does not require a JNDI implementation. For example, Oracle Advanced Queuing facility (Oracle AQ). When selected the applicable fields display.

Class name

The name of the Java class for implementing the connection to the message queue. A Java class for Oracle AQ is available. The class name is:

```
com.cyclonecommerce.tradingengine.transport.jms.OracleAQ
QueueUtil
```

If you want a Java class for a provider other than Oracle AQ, you need the help of a professional services consultant. Contact technical support for information.

Parameters

These are the parameters for implementing the Java class. There are four parameters required for the Java class for Oracle AQ. These parameters must be in the following order:

Host. The name of the computer running Oracle AQ.

Queue name. The name of the Oracle AQ message queue.

Port. The port Oracle AQ uses to listen for messages.

Driver type. The type of JDBC driver for connecting to the provider. For Oracle AQ, the valid values are **thin** and **oci8**.

Advanced tab (JMS polled)

Maximum concurrent connections

If the system uses multiple JVM nodes, the maximum number of connections the system can make to this exchange at the same time. If you change this field, the number should at least equal the number of nodes.

Retries

This is the number of times the trading engine will retry connecting to the partner's transport if the initial attempt to connect and send the message failed. The following are common reasons for triggering retries.

1. The connection attempt failed immediately for a reason such as host not found.
2. The host was found, but the connection process took longer than the connect timeout interval specified on the Advanced tab.
3. The connection was successful, but the partner's HTTP server took longer than the response timeout interval to return a 200 OK response indicating the message was successfully received. A 200 OK response is a transport response, separate from a message protocol response such as an AS2 receipt.

Note that in the last case, the 200 OK response also will include the receipt if synchronous receipts were requested. Otherwise, it will be a simple 200 OK response with no payload. And if an asynchronous receipt was requested, the partner will connect later to send it.

Retries occur according to an algorithm that starts at 5 minutes. The interval between retries increases with each subsequent retry in this pattern: 10 minutes, 15 minutes, 30 minutes, 60 minutes. The interval plateaus at 60 minutes. This means if the retry value is greater than 5, the fifth and each subsequent retry occurs at 60 minute intervals.

For example, if retries is 3, the system will try connecting again in 5 minutes if the initial connection attempt fails. If this retry attempt also fails, the system attempts a second retry in 10 minutes. The third retry attempt is made 15 minutes later. If the third retry attempt fails, the message is given a failed status. So after four attempts (the first attempt plus 3 retries), the message fails. You can search for and manually resubmit failed messages in Message tracker.

Retries do not occur precisely at these intervals because each connection attempt takes some seconds, which varies by computer. So retries actually occur after the connection attempt time plus the interval.

This control applies only to retrying to send messages, not receiving. It applies only to retrying to send related to transport issues. It does not apply to successfully sent messages for which receipts have not been received as expected. Another control, resends, determines how many times the system will resend a message when a receipt is not received from the partner. For information about resends, see reliable messaging in the collaboration settings chapter.

Message Type

Specifies the JMS message class.

BytesMessage. A BytesMessage object is used to send a message containing a stream of uninterpreted bytes. It inherits from the Message interface and adds a bytes message body.

TextMessage. A TextMessage object is used to send a message containing a java.lang.String. It inherits from the Message interface and adds a text message body.

For more information about Java classes see http://java.sun.com/products/jms/docs.html.

This field applies only to JMS when used for receiving messages from partners or integration delivery.

Back up the files that go through this transport

Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

Use transacted queue

Select this if the provider is Oracle AQ. Otherwise, do not select it.

Maximum files per polling interval

The highest number of messages the system can retrieve each time it polls.

Polling interval (seconds)

The interval in seconds the system waits before polling for messages to retrieve.

Post-processing script

The full path to an executable file that contains post-processing commands. This field is available for community delivery integration exchanges and partner trading delivery exchanges.

Advanced tab (JMS listener)

JMS Server reconnect interval (seconds)

Specifies the interval for re-establishing a connection to the JMS server if the server goes down.

This field applies only to JMS when used as an asynchronous integration pickup transport.

Back up the files that go through this transport

Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

MQSeries maintenance

The following topics document the fields on the maintenance pages for the MQSeries transport.

The maintenance pages display the transport fields that can be changed, while the delivery exchange wizard has only the most commonly used. The following are the fields on the settings and advanced tabs. For information about fields on the other maintenance tabs, see the topics for the specific tabs.

IBM MQSeries settings tab

MQSeries server

The fully qualified domain name or IP address of the MQSeries host.

Port

The port where the application listens for incoming documents. The default is 1414.

Queue name

The name of the MQSeries queue that receives incoming documents.

Queue manager

The name of the MQSeries queue manager.

Channel

The name of the communications channel.

Characters set

The character set used by the queue manager. This number should match the number used by the queue manager. The default is 819.

This server requires a user name and password

Select this check box to supply a user name and password for basic authentication. Clear this check box if the server does not require basic authentication.

User name

The user name to connect to the server.

Password

The password to connect to the server.

Confirm password

The password entered again for confirmation.

Advanced tab

Maximum concurrent connections

If the system uses multiple JVM nodes, the maximum number of connections the system can make to this exchange at the same time. If you change this field, the number should at least equal the number of nodes.

Retries

This is the number of times the trading engine will retry connecting to the partner’s transport if the initial attempt to connect and send the message failed. The following are common reasons for triggering retries.

1. The connection attempt failed immediately for a reason such as host not found.
2. The host was found, but the connection process took longer than the connect timeout interval specified on the Advanced tab.

3. The connection was successful, but the partner's HTTP server took longer than the response timeout interval to return a 200 OK response indicating the message was successfully received. A 200 OK response is a transport response, separate from a message protocol response such as an AS2 receipt.

Note that in the last case, the 200 OK response also will include the receipt if synchronous receipts were requested. Otherwise, it will be a simple 200 OK response with no payload. And if an asynchronous receipt was requested, the partner will connect later to send it.

Retries occur according to an algorithm that starts at 5 minutes. The interval between retries increases with each subsequent retry in this pattern: 10 minutes, 15 minutes, 30 minutes, 60 minutes. The interval plateaus at 60 minutes. This means if the retry value is greater than 5, the fifth and each subsequent retry occurs at 60 minute intervals.

For example, if retries is 3, the system will try connecting again in 5 minutes if the initial connection attempt fails. If this retry attempt also fails, the system attempts a second retry in 10 minutes. The third retry attempt is made 15 minutes later. If the third retry attempt fails, the message is given a failed status. So after four attempts (the first attempt plus 3 retries), the message fails. You can search for and manually resubmit failed messages in Message tracker.

Retries do not occur precisely at these intervals because each connection attempt takes some seconds, which varies by computer. So retries actually occur after the connection attempt time plus the interval.

This control applies only to retrying to send messages, not receiving. It applies only to retrying to send related to transport issues. It does not apply to successfully sent messages for which receipts have not been received as expected. Another control, resends, determines how many times the system will resend a message when a receipt is not received from the partner. For information about resends, see reliable messaging in the collaboration settings chapter.

Back up the files that go through this transport

Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

Maximum files per polling interval

The highest number of messages the system can retrieve each time it polls.

Polling interval (seconds)

The interval in seconds the system waits before polling for messages to retrieve.

Post-processing script

The full path to an executable file that contains post-processing commands. This field is available for community delivery integration exchanges and partner trading delivery exchanges.

Web services API maintenance

The following topics document the fields on the maintenance pages for the web services API transport.

The maintenance pages display the transport fields that can be changed, while the delivery exchange wizard has only the most commonly used. The following are the fields on the settings and advanced tabs. For information about fields on the other maintenance tabs, see the topics for the specific tabs.

Web services API client settings tab

URL

The URL the trading engine uses to post messages to a back-end system for integration delivery.

Advanced (Web services API client) tab

Maximum concurrent connections

If the system uses multiple JVM nodes, the maximum number of connections the system can make to this exchange at the same time. If you change this field, the number should at least equal the number of nodes.

Retries

This is the number of times the trading engine will retry connecting to the partner's transport if the initial attempt to connect and send the message failed. The following are common reasons for triggering retries.

1. The connection attempt failed immediately for a reason such as host not found.
2. The host was found, but the connection process took longer than the connect timeout interval specified on the Advanced tab.
3. The connection was successful, but the partner's HTTP server took longer than the response timeout interval to return a 200 OK response indicating the message was successfully received. A 200 OK response is a transport response, separate from a message protocol response such as an AS2 receipt.

Note that in the last case, the 200 OK response also will include the receipt if synchronous receipts were requested. Otherwise, it will be a simple 200 OK response with no payload. And if an asynchronous receipt was requested, the partner will connect later to send it.

Retries occur according to an algorithm that starts at 5 minutes. The interval between retries increases with each subsequent retry in this pattern: 10 minutes, 15 minutes, 30 minutes, 60 minutes. The interval plateaus at 60 minutes. This means if the retry value is greater than 5, the fifth and each subsequent retry occurs at 60 minute intervals.

For example, if retries is 3, the system will try connecting again in 5 minutes if the initial connection attempt fails. If this retry attempt also fails, the system attempts a second retry in 10 minutes. The third retry attempt is made 15 minutes later. If the third retry attempt fails, the message is given a failed status. So

after four attempts (the first attempt plus 3 retries), the message fails. You can search for and manually resubmit failed messages in Message tracker.

Retries do not occur precisely at these intervals because each connection attempt takes some seconds, which varies by computer. So retries actually occur after the connection attempt time plus the interval.

This control applies only to retrying to send messages, not receiving. It applies only to retrying to send related to transport issues. It does not apply to successfully sent messages for which receipts have not been received as expected. Another control, resends, determines how many times the system will resend a message when a receipt is not received from the partner. For information about resends, see reliable messaging in the collaboration settings chapter.

Send the entire payload contents

Send the payload through this transport. This option is only for integration delivery.

Send the payload URL only

Send only the URL that points to the payload and not the payload itself. See Integration delivery on page 221 for conditions on using this option. This option is only for integration delivery.

Back up the files that go through this transport

Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

Post-processing script

The full path to an executable file that contains post-processing commands. This field is available for community delivery integration exchanges and partner trading delivery exchanges.

Web services API server settings tab

Embedded web services API server

click the link to display the settings for the global web services API server.

Advanced tab (web services API server)

Back up the files that go through this transport

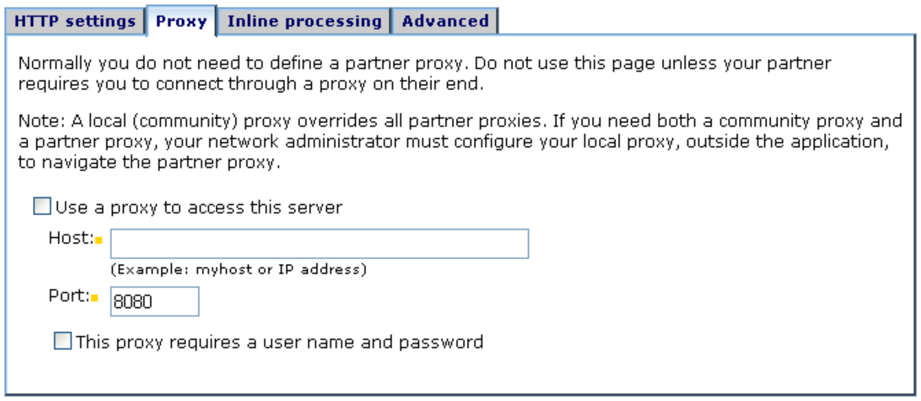
Indicates whether the system backs up copies of the messages it retrieves from integration or receives from partners.

Backing up files is recommended. This is required for the system to perform fail-over operations such as attempting to send messages again (retries) in case of a transport connection failure. Backups also are needed if you want the ability to resubmit messages to back-end integration or resend messages to partners.

Backup files are stored in \[install directory]\[build number]\common\data\backup, unless you specify otherwise.

Proxy tab

The delivery exchange wizard and maintenance pages for HTTP and HTTPS transports for partners have fields for specifying whether connections must be made through a proxy. The following describes the fields for the proxy tab on the maintenance page.



Normally you do not need to define a partner proxy. Do not use this page unless your partner requires you to connect through a proxy on their end.

Note: A local (community) proxy overrides all partner proxies. If you need both a community proxy and a partner proxy, your network administrator must configure your local proxy, outside the application, to navigate the partner proxy.

☐ Use a proxy to access this server

Host:

(Example: myhost or IP address)

Port: 8080

☐ This proxy requires a user name and password

Figure 56. Proxy tab on transport maintenance page

Use a proxy to access this server

Select this check box if you must pass through a proxy to reach the delivery exchange server. For partner exchanges, the proxy would be between the trading engine and the partner's HTTP server.

Host

The URL of the proxy host.

Port

The port where the proxy host listens for connections.

This proxy requires a user name and password

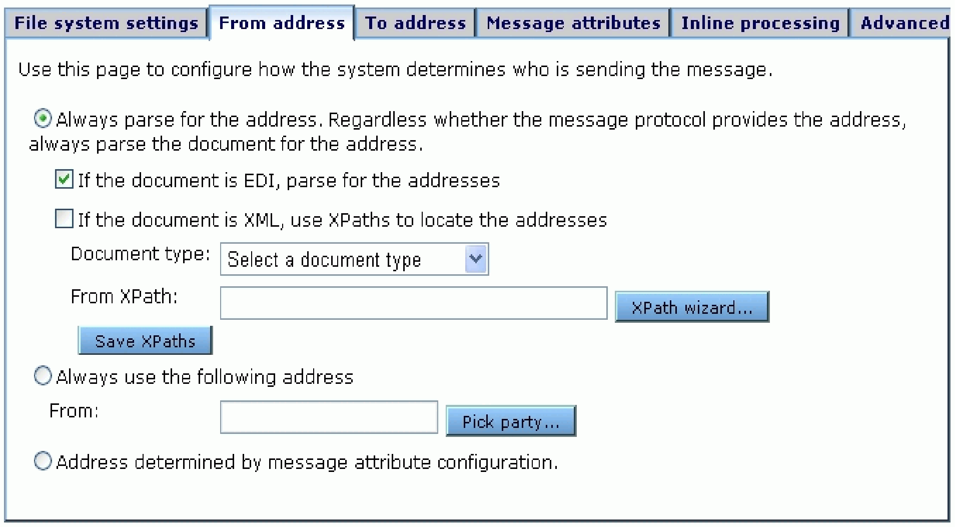
Select this check box if the proxy requires a user name and password before it accepts the connection. Obtain this information from your partner.

From address and To address tabs

Delivery exchanges allow you to specify parsing rules for the documents it consumes. You can specify parsing for sender or receiver or both, or you can specify the sender and receiver routing IDs to use in all cases.

You can specify addressing information on the maintenance pages for a community trading delivery exchange and integration pickup delivery exchange, as well as in the delivery exchange wizard when adding a pickup integration delivery exchange.

The user interface uses separate maintenance tabs to specify parsing rules for sender and receiver. The fields are used the same way for both tabs, so the following field descriptions apply to both.



Use this page to configure how the system determines who is sending the message.

(●) Always parse for the address. Regardless whether the message protocol provides the address, always parse the document for the address.

[✓] If the document is EDI, parse for the addresses

[ ] If the document is XML, use XPaths to locate the addresses

Document type: Select a document type

From XPath:

( ) Always use the following address

From:

( ) Address determined by message attribute configuration.

Figure 57. From address tab on transport maintenance page

Always parse for the address

Specifies that the trading engine should always parse the message for the sender or receiver address.

If the document is EDI, parse for the addresses

If an EDI document is picked up, use the “to” and “from” addresses specified within it. Properly formatted EDI documents contain this information.

If the document is XML, use XPaths to locate the addresses

If an XML document is picked up, use the “to” and “from” addresses specified by the XPaths within it. XML Path Language or XPath is a language for addressing parts of an XML document.

Document type

Choose the XML type and the system provides the values in the **From XPath** and **To XPath** fields. If you use another XML type, click **XPath wizard** and use the wizard to specify the XPaths using your example of the XML document. You can use the XPath wizard for the “from” or “to” address or both. Using the XPath wizard requires knowledge of XML.

From/To XPath

The XPath for the message sender or receiver, depending on the tab you are using.

Always use the following address

Specifies that the trading engine should always use a fixed address for the sender or receiver. You can click **Pick party** to launch a wizard that helps you locate the community or partner you want. The “from” or “to” party must be set up as a community or partner in the system.

Address determined by message attribute configuration

Select this if the “from” or “to” address or both is configured using the Message attributes tab tab. This option is available only for the file system transport.

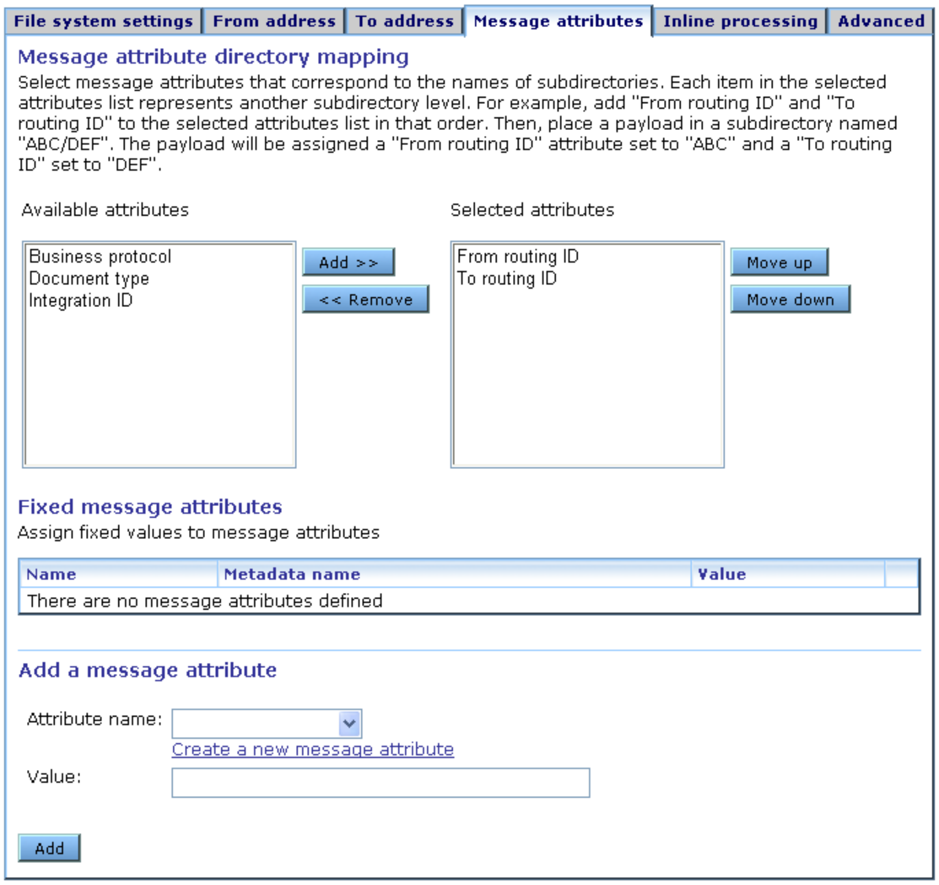
Message attributes tab

You can configure exchanges to attach meta-data to messages the trading engine picks up from integration or receives from partners. Meta-data are information about documents, such as the sender and receiver routing IDs and many other attribute elements.

There are two methods for attaching meta-data: directory mapping and fixed message attributes. Directory mapping is available for file system exchanges for integration and receiving messages from partners. Fixed message attributes are available for all exchanges for integration and receiving messages from partners.

The message attributes tab on exchange maintenance pages is used for configuring directory mapping, fixed message attributes or both. The following topics describe how to use each method.

- Directory mapping
- Fixed message attributes on page 297



Message attribute directory mapping

Select message attributes that correspond to the names of subdirectories. Each item in the selected attributes list represents another subdirectory level. For example, add "From routing ID" and "To routing ID" to the selected attributes list in that order. Then, place a payload in a subdirectory named "ABC/DEF". The payload will be assigned a "From routing ID" attribute set to "ABC" and a "To routing ID" set to "DEF".

Available attributes: Business protocol, Document type, Integration ID

Selected attributes: From routing ID, To routing ID

Fixed message attributes

Assign fixed values to message attributes

Name	Metadata name	Value
There are no message attributes defined		

Add a message attribute

Attribute name:

Create a new message attribute

Value:

Figure 58. Message attributes tab

Directory mapping

Use this procedure to map meta-data attributes to file system directories. The names of the directories are used as attribute values. This is useful when you want an integration file system to control meta-data about outbound documents.

The common use of meta-data directory mapping is for file system pickup and delivery integration exchanges. It also can be used for file system exchanges for receiving messages from partners. As file system rarely is employed as a transport for receiving messages, directory mapping is described in the context of file system integration.

Setting up file system integration exchanges for directory mapping is a two-step process that can be completed in any order:

1 In the user interface specify meta-data attributes. You can choose a default attribute or define your own.

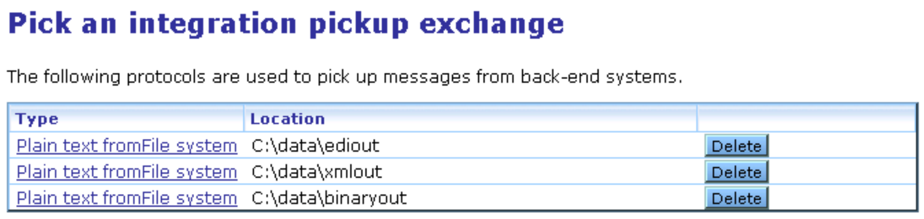
2 For integration pickup, create subdirectories in the file system that correspond to the order of the selected attributes. For integration delivery, the trading engine creates the directories for you. If an attribute value contains an unsupported character, the trading engine uses the hexadecimal value instead. For example, for **application/ pdf**, the directory name is **application%2fpdf**, with “%2f” used in place of “/”.

The following steps provide more configuration details.

Steps

1 Determine the attribute values you want the file system to set for messages. For example, the sender and receiver’s routing IDs.

2 Click **Integration pickup** or **Integration delivery** on the navigation graphic at the top of the community summary page.



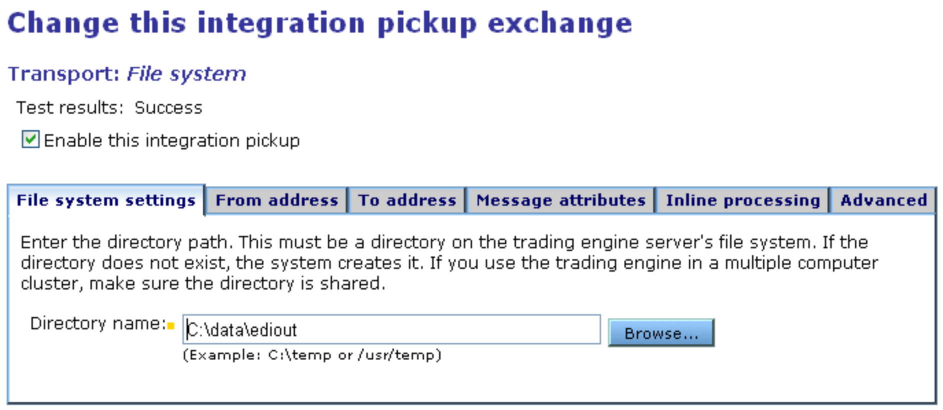
Pick an integration pickup exchange

The following protocols are used to pick up messages from back-end systems.

Type	Location	
Plain text fromFile system	C:\data\ediout	Delete
Plain text fromFile system	C:\data\xmlout	Delete
Plain text fromFile system	C:\data\binaryout	Delete

Figure 59. Integration pickup exchanges

3 Click the name of an integration exchange to open its maintenance page.



Change this integration pickup exchange

Transport: *File system*

Test results: Success

☑ Enable this integration pickup

Enter the directory path. This must be a directory on the trading engine server's file system. If the directory does not exist, the system creates it. If you use the trading engine in a multiple computer cluster, make sure the directory is shared.

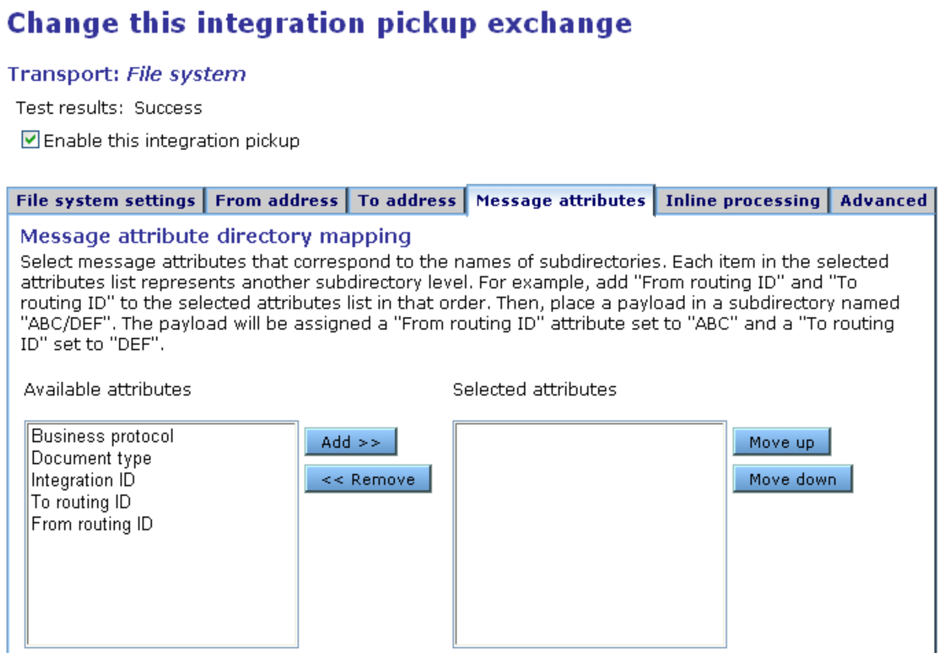
Directory name: C:\data\ediout Browse...

(Example: C:\temp or /usr/temp)

Figure 60. File system settings tab for integration pickup exchange

Note the directory name on the file system settings tab. You are going to specify attributes corresponding to subdirectories of this directory. For integration pickup, you will create the subdirectories in the file system later.

4 Select the message attributes tab.


Change this integration pickup exchange

Transport: *File system*

Test results: Success

☑ Enable this integration pickup

Message attribute directory mapping

Select message attributes that correspond to the names of subdirectories. Each item in the selected attributes list represents another subdirectory level. For example, add "From routing ID" and "To routing ID" to the selected attributes list in that order. Then, place a payload in a subdirectory named "ABC/DEF". The payload will be assigned a "From routing ID" attribute set to "ABC" and a "To routing ID" set to "DEF".

Available attributes: Business protocol, Document type, Integration ID, To routing ID, From routing ID

Selected attributes:

Figure 61. Directory mapping section of message attributes tab

The top half of the tab concerns directory mapping and the bottom half concerns fixed attributes. This procedure only covers directory mapping. Fixed attributes are covered in Fixed message attributes on page 297

You can use both directory mapping and fixed attributes, but make sure the attributes do not overlap. If overlapping occurs, a fixed attribute takes precedence over directory mapping.

5 Select an available attribute and use the **Add** button to move it to the selected list. The order of attributes is important, as each attribute is matched to a subdirectory name in the directory hierarchy.

For example, if you select **From routing ID** and **To routing ID** and the file system directory name is c:\data\ediout, the trading engine will expect to pick up messages in the lowest directory in the following path:

c:\data\ediout\[from routing ID]\[to routing ID]

6 If the attribute you want is not listed, do the following to add it.

Click the **Create a new message attribute** link near the bottom of the tab. Type the attribute name and click **Add**. When you return to the message attributes tab, the attribute appears in the available attribute list.

7. Click **Save changes** at the bottom of the message attributes tab to save the selected attributes.

8. For integration pickup, if you selected **From routing ID** or **To routing ID,** as we did in our example, select the from address tab or to address tab or both as applicable. At the bottom of the tab select **Address determined by message attribute configuration** and click **Save changes**.

9. For integration pickup, create directories in the file system that correspond to the selected attributes. For example, if From routing ID is ZZworldwide and To routing ID is ZZacme and the file system directory is c:\datat\ediout, you would create subdirectories matching the following structure.

   c:\data\ediout\ZZworldwide\ZZacme.

   When the trading engine picks up a document from the ZZacme subdirectory, it will assign the sender ID as ZZworldwide and the receiver ID as ZZacme.

   The directory structure is scalable for a community with multiple partners. For example, the directory structure could match the following.

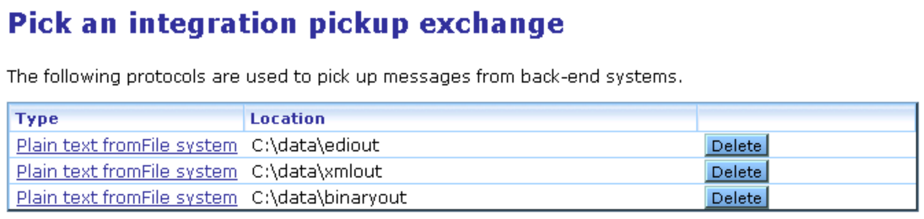
c:\	data\	ediout\	ZZworldwide\	ZZacme
				routing ID 2
				routing ID 3
				and so on

Fixed message attributes

Use this procedure to associate attributes and fixed values to documents in the configuration of exchanges used to receive messages from partners or retrieve messages from back-end systems. Whenever the exchange handles a document, the meta-data are attached. This is useful when, for instance, you want to trigger processing actions based on an attribute or engage in ebXML trading using static meta-data.

Steps

1 Click **Integration pickup** on the navigation graphic at the top of the community summary page.



Pick an integration pickup exchange

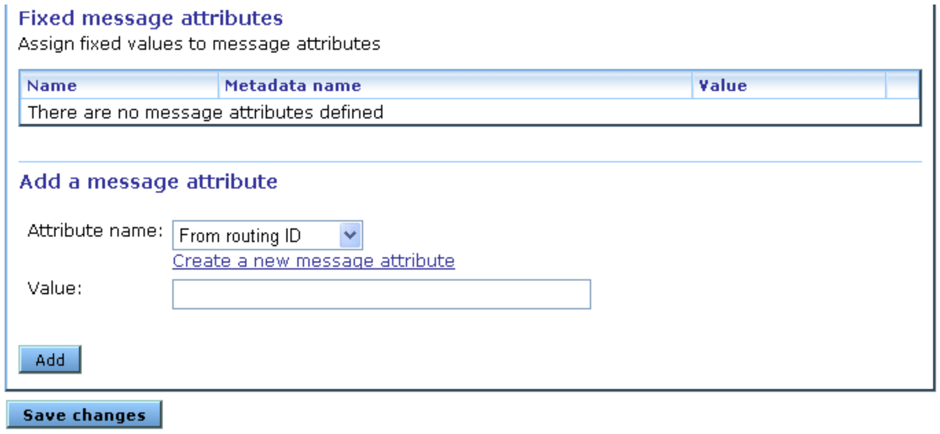
The following protocols are used to pick up messages from back-end systems.

Type	Location	
Plain text fromFile system	C:\data\ediout	Delete
Plain text fromFile system	C:\data\xmlout	Delete
Plain text fromFile system	C:\data\binaryout	Delete

Figure 62. Integration pickup exchanges

2 Click the name of an integration pickup exchange to open its maintenance page.

3 Select the message attributes tab.



Fixed message attributes

Assign fixed values to message attributes

Name	Metadata name	Value	
There are no message attributes defined			

Add a message attribute

Attribute name: From routing ID

Create a new message attribute

Value:

Figure 63. Fixed message attributes section of message attributes tab

The top half of the tab concerns directory mapping and the bottom half concerns fixed message attributes. This procedure only covers fixed message attributes. Directory mapping is covered in Directory mapping on page 294

You can use both directory mapping and fixed attributes, but make sure the attributes do not overlap. If overlapping occurs, a fixed attribute takes precedence over directory mapping.

4 Select an attribute name and type a value and click **Add**. To add more attributes and values, repeat the step. Once you have added an attribute and fixed value, you can delete the name-value pair, but you cannot change it.

The following describes the fields.

Attribute name

Select the name of an attribute. The attribute and its value could trigger, for instance, a post-processing action. Also, for example, if you want to do ebXML trading without using MMDs, you would add attributes matching the required MMD elements. See Post-processing message meta-data on page 223 or Using fixed meta-data on page 457.

If the attribute you want is not listed, do the following to add it.

Click the **Create a new message attribute** link near the bottom of the tab. Type the attribute name and click **Add**. When you return to the message attributes tab, the attribute appears in the attribute name list.

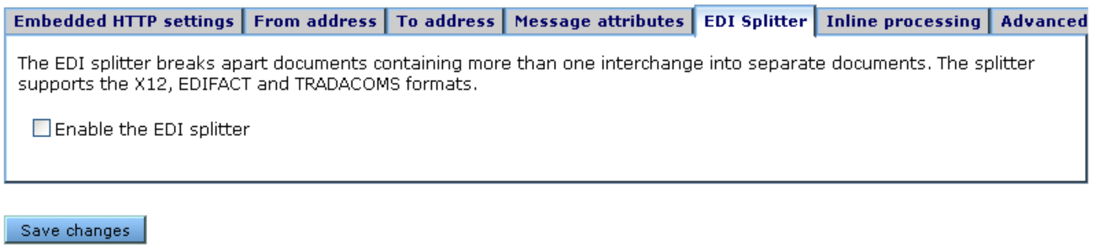
Value

The fixed value of the attribute. This can be a number or text.

5 Click **Save changes** to save the attributes and values you have added.

EDI Splitter tab

Batch EDI documents can be split into individual documents. Splitting can be enabled in transports for retrieving documents from integration or receiving documents from partners. The only configuration needed to enable splitting is selecting the **Enable the EDI splitter** check box on the EDI Splitter tab of a transport's maintenance page.



The EDI splitter breaks apart documents containing more than one interchange into separate documents. The splitter supports the X12, EDIFACT and TRADACOMS formats.

☐ Enable the EDI splitter

Figure 64. EDI Splitter tab

The EDI splitter works for X12, EDIFACT, and TRADACOMS documents. The splitter looks for segments bracketed between interchange control headers. These are:

EDI document type	Header segment	Trailer segment
X12	ISA	IEA
EDIFACT	UNB	UNZ
TRADACOMS	STX	END

The splitter rejects documents whose control numbers do not match or are missing a trailer segment.

This integrated utility splits interchanges, including the headers and trailers, into separate documents containing a single interchange. If the original file contains only one interchange, the utility produces only one file, minus any extraneous data following the trailer segment.

In Message tracker the original unsplit document has a status of **Split**. When viewing the details of the original, the document summary tab reports “Message split” and provides links to the split payloads. When viewing the details of a split payload, the document summary tab reports “Message is the result of a split” and provides a link to the unsplit original.

Inline processing tab

The extensible architecture of the trading engine allows system integrators to apply custom logic to in-process messages as an integral part of the processing pipeline. The custom processing logic, implemented as a user-defined Java class, can be selectively applied at runtime to inbound or outbound messages.

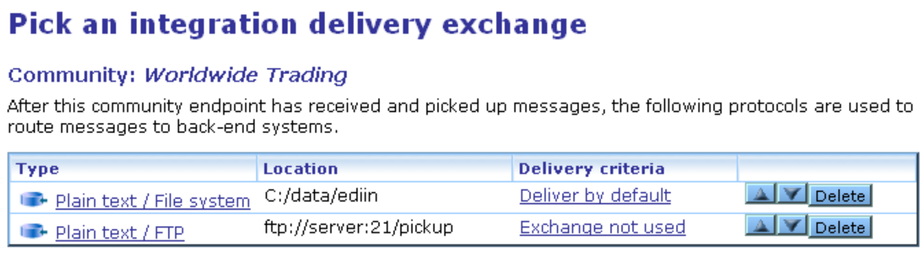
Use of this feature requires obtaining an optional developer’s license.

Delivery criteria tab

Use this procedure to set rules that direct inbound messages to certain integration delivery exchanges based on rules. The delivery criteria tab on the maintenance pages of integration delivery exchanges is used for this purpose.

Steps

1 Click **Integration delivery** on the navigation graphic at the top of the community summary page.



Pick an integration delivery exchange

Community: *Worldwide Trading*

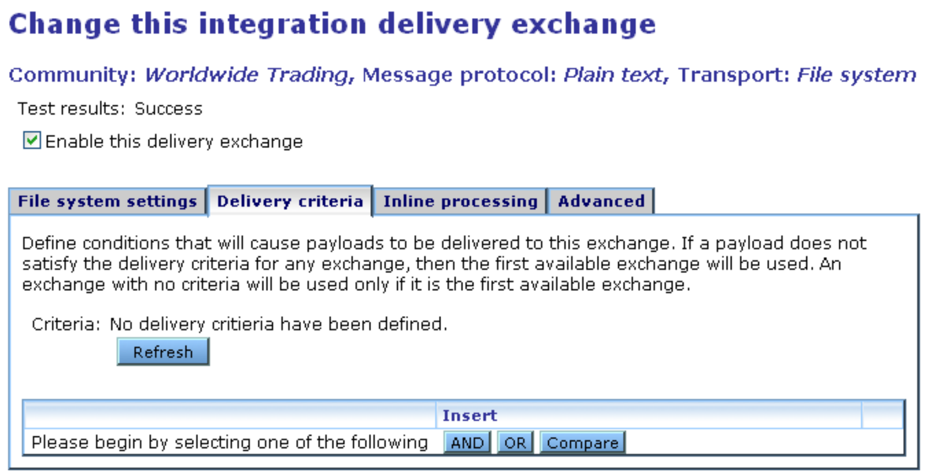
After this community endpoint has received and picked up messages, the following protocols are used to route messages to back-end systems.

Type	Location	Delivery criteria	
Plain text / File system	C:/data/ediin	Deliver by default	Delete
Plain text / FTP	ftp://server:21/pickup	Exchange not used	Delete

Figure 65. List of integration delivery exchanges for a community

2 Click the name of an integration delivery exchange to open its maintenance page.

3 Select the delivery criteria tab.



Change this integration delivery exchange

Community: *Worldwide Trading*, Message protocol: *Plain text*, Transport: *File system*

Test results: Success

☑ Enable this delivery exchange

Define conditions that will cause payloads to be delivered to this exchange. If a payload does not satisfy the delivery criteria for any exchange, then the first available exchange will be used. An exchange with no criteria will be used only if it is the first available exchange.

Criteria: No delivery critieria have been defined.

Please begin by selecting one of the following

Figure 66. Delivery criteria tab

4 You can set up a single or multiple conditions for directing inbound messages to this integration delivery exchange. Review the following guideline on what to do next:

If one condition, click **Compare**

If multiple conditions and all must be met, click **AND**

If multiple conditions and at least one must be met, click **OR**

5 Click **Compare**, **AND** or **OR** depending on how many conditions you want.

If you click **Compare**, the tab refreshes, displaying fields for attribute, operator and value. Select an attribute from the list, an operator and a value. For example, **Business protocol equals EDIINT AS1**. Click **Save Changes**.

If you click **AND** or **OR**, the tab refreshes. Click **Compare** and the tab refreshes again, displaying fields for attribute, operator and value. Select an attribute, operator and value and click **Compare** again. Select another attribute, operator and value and click **Save Changes**.

If the attribute you want is not listed, click **Message handler** on the navigation graphic at the top of the community summary page. Click **Add a message attribute definition** at the bottom of the message handler page. Type the attribute name and click **Add**. When you return to the delivery criteria tab, the attribute appears in the available attribute list.

If you choose Content MIME type as an attribute, the following are commonly used types.

application/EDI-consent	Tradacoms messages
application/EDIFACT	EDIFACT messages
application/EDI-X12	X12 messages
application/octet-stream	Binary messages
application/xml	XML messages

Change embedded transports

Use this procedure after setting up an embedded SMTP or HTTP transport to change its settings. The embedded transport maintenance page is accessible from a community summary page. Because embedded transport servers can be shared between different delivery exchanges, changes you make to the servers are reflected in all delivery exchanges that use them.

Steps

1. From the community summary page, click **Change an embedded transport server** at the bottom of the page. A list of embedded transports displays.
2. Click the embedded transport server you want to change.
3. Edit the values on the Settings or Advanced tabs as necessary, and then click **Save changes**.

See also:

- Embedded e-mail for community on page 192

- Embedded HTTP transport on page 195
- Embedded transport servers on page 167

❖ ❖ ❖

16 Outbound HTTP proxy

The trading engine lets you define a global HTTP proxy through which all outbound HTTP traffic is routed. All communities use this proxy.

If your infrastructure does not require HTTP traffic to navigate a proxy to access the Internet, you do not need to use this feature.

Note: Proxies are “one-hop” by nature, so if you set up a global HTTP proxy, it overrides any other proxy definitions that your partners may have set up. If you know that one or more of your partners use a proxy for incoming HTTP connections, you should not use a global HTTP proxy or work with your partner to establish some other connection method.

To set up the HTTP proxy, click the HTTP proxy area of the navigation graphic on the community summary page

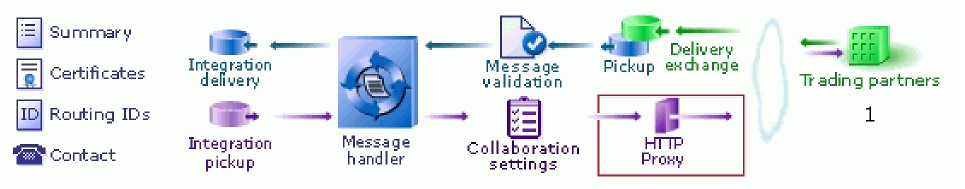
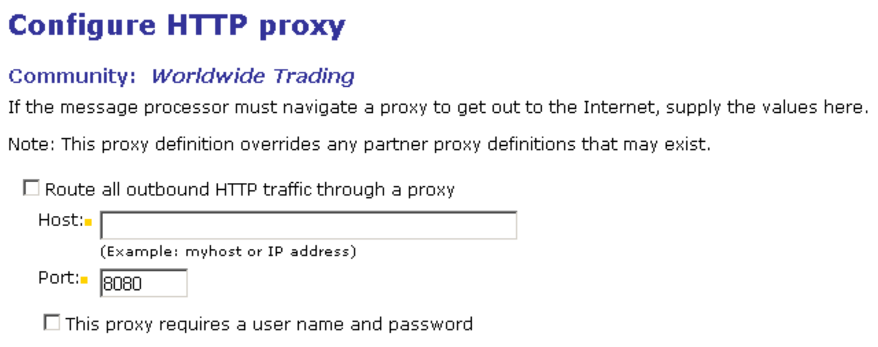


Figure 67. HTTP proxy on the navigation graphic



Configure HTTP proxy

Community: *Worldwide Trading*

If the message processor must navigate a proxy to get out to the Internet, supply the values here.

Note: This proxy definition overrides any partner proxy definitions that may exist.

☐ Route all outbound HTTP traffic through a proxy

Host:

(Example: myhost or IP address)

Port: 8080

☐ This proxy requires a user name and password

Figure 68. Outbound HTTP proxy configuration

The following describes the proxy fields.

Route all outbound HTTP traffic through a proxy

Select this check box to use a global HTTP proxy. Clear this check box if you do not want to use a global HTTP proxy.

Host

The fully qualified domain name or IP address of the HTTP proxy.

Port

The port through which outbound HTTP traffic is routed.

This proxy requires a user name and password

Select this check box to supply a user name and password for basic authentication. Clear this check box if your proxy does not require basic authentication.

User name

The user name to connect to the server.

Password

The password to connect to the server.

Confirm password

The password entered again for confirmation.

❖ ❖ ❖

17 Certificates and keys

The trading engine offers true security by providing authentication, confidentiality, integrity and non-repudiation of documents. The trading engine uses state-of-the-art cryptography to ensure the security of the documents you exchange over the public Internet.

For a glossary of Internet security terms, go to http://www.ietf.org/rfc/rfc2828.txt.

Concepts

PKI description

The trading engine supports public key infrastructure (PKI) to securely trade business documents over the Internet. PKI is a system of components that use digital certificates and public key cryptography to secure transactions and communications.

PKI uses certificates issued by certificate authorities (CAs) to provide authentication, confidentiality, integrity and non-repudiation of data. The following defines these in more detail.

Authentication	Authentication is verification of the identity of a person or process. Authentication confirms that a message truly came from the source that sent it.
Confidentiality	Confidentiality is the assurance that a message has been disclosed only to the parties authorized to share the information.

Integrity	Integrity is the assurance that the information has not been altered in any way and is precisely true to the source.
Non-repudiation	Non-repudiation is proof that a recipient received a message. This protects a sender from a false denial that a recipient did not receive a message.

PKI options

There are two PKI options, and the trading engine supports both. They are self-signed certificates and commercial PKIs. The option you choose can depend on a number of factors, such as cost, human and system resources, and the degree or sophistication of security desired.

Self-signed certificates generated by the trading engine and certificates generated by commercial PKIs all support the X.509 standard for public key certificates. You can use any X.509 certificate, regardless of the source, in document transactions with partners. For example, you can generate a self-signed certificate for your community profile and export a public encryption key in a certificate with the profile to a partner for use in encrypting and signing documents sent to you. Meanwhile, you can engage in trading with partners who have sent you public keys in Entrust or VeriSign certificates.

The following explains each security option in more detail.

Self-signed certificates

The trading engine can generate root certificates in which you are, in effect, acting as your own certificate authority. The trading engine supports single-key pair self-signed certificates for both encrypting and signing documents and dual-key pair self-signed certificates in which one certificate is used for encrypting and the other for signing.

Self-signed certificates are easy to make and use. They are best suited for use within relatively small trading groups. This is because you must implicitly trust a partner’s self-signed certificate; there is no chain of trust to independently vouch for the certificate. Such a trust relationship can more suitably be managed among a small number of partners.

Although self-signed certificates can provide a high-degree of security, the degree depends on the vigilance and administrative skills of the persons managing them. Generally speaking, the use of self-signed certificates does not have the rigorous discipline and orderly structure inherent to a commercial PKI.

Commercial PKIs

A commercial PKI is an organization set up for the centralized creation, distribution, tracking and revocation of keys for a potentially large community of partners. A commercial PKI has a documented certificate policy (CP) that indicates the applicability of a public key certificate to a specific community or class of applications with common security requirements. A commercial PKI also has a certification practice statement (CPS), which details the practices the CA follows for issuing public key certificates.

There are two types of commercial PKIs:

In house	An in-house PKI enables you to achieve complete control of security policies and procedures, but also carries the burden of management and cost to set up and maintain the system.
Outsourced	You can leverage the services of PKI systems such as VeriSign, Baltimore and other third-party certificate authorities. You purchase keys and certificates for use in trading partner relationships and let the CA manage security policies and such details as certificate revocation. The level of outsourcing can range from purchasing an end-entity public key certificate of a certain validity period from a commercial PKI to outsourcing all of the PKI services that your organization requires.

The role of trust in PKI

PKI establishes digital identities that can be trusted. The CA is the party in a PKI responsible for certifying identities. More than generating a certificate, this entails verifying the identity of a subscriber according to established policies and procedures. This is the case for in-house and outsourced PKIs. In an organization that generates and uses its own self-signed certificates, the trading parties must verify the certificates and establish a direct trust. Once established that an identity or issuer of an identity can be trusted, the trust anchor’s certificate is stored in a local trust list.

The trading engine has a local trust list for storing and managing established trust relationships (Trusted roots certificates tab on page 327). The application maintains a list of common public CA certificates similar to those kept in web browsers. Although convenient, this pre-determination of trust might not complement your organization’s security policy. The decision of who to trust rests with your organization. For example, a trader might accept certificates issued by its own root CA and its trading partners' root CA, but not from partner B, who the trader has not done business with in the past. If you choose not to accept partner B's root CA certificate, your system will not accept any certificates issued by partner B. The greater the number of root CA certificates you choose to accept, the more open your community is to others.

Scalability

The use of self-signed certificates relies on users to exchange certificates and establish trust in each other. This informal web of trust works for small groups, but can become unmanageable for large numbers of partners. In contrast, an in-house or outsourced PKI uses hierarchies, where a certificate authority serves as a trust anchor for many users. Once trust has been established for the certificate authority, it is unnecessary to re-establish the trust for other certificates the CA issues. Establishing hierarchies of users scales equally well for small and large groups.

Certificate revocation

A certificate is expected to be usable for its entire validity period. However, there are circumstances when a certificate should no longer be considered valid even though it has not expired. Possible circumstances range from a user name change to suspected compromise of the private key. In such circumstances an in-house or outsourced CA can revoke the certificate. The trading engine can be configured to compare your partners' certificates against lists of revoked certificates issued by CAs. However, self-signed certificates cannot be revoked. You must notify all partners using the certificate that it should no longer be trusted.

Dual-key pairs

Support for two pairs of public-private keys is a fundamental requirement for some PKIs (for example, Entrust). One key pair is for data encryption and the other key pair is for digitally signing documents. Encryption key pairs and signing key pairs are a result of conflicting requirements. One such requirement is to support different algorithms for encryption and digital signature pairs and different validity periods. Another reason is to support data recovery, which requires the private keys for decrypting to be securely backed up, but non-repudiation, which requires the private keys for signing, not to be backed up. There also might be the requirement to support updating encryption key pairs and managing decryption key histories even though this conflicts with the requirement to securely destroy the private key used for signing when updating signing key pairs. Using two key pairs — an encryption key pair and signing key pair — solves these conflicting requirements.

Why use encryption and digital signatures

Encrypting and digitally signing documents by using certificates provides the following assurances about document transmissions:

- Only the addressee can read the message and not any unauthorized people. Encryption provides this assurance.
- The message cannot be tampered with. That is, data cannot be changed, added or deleted without you knowing it. A document's digital signature provides this assurance.
- Partners who send you documents are genuinely who they claim to be. Likewise, when partners receive documents signed by you, they can be confident the documents came from you. A document's digital signature provides this assurance.
- The partners who send you documents cannot claim they did not send them. This is referred to as non-repudiation of origin. A document's digital signature provides this assurance.
- Partners to whom you send documents cannot claim they did not receive them. This is referred to as non-repudiation of receipt. A signed document acknowledgment provides this assurance.

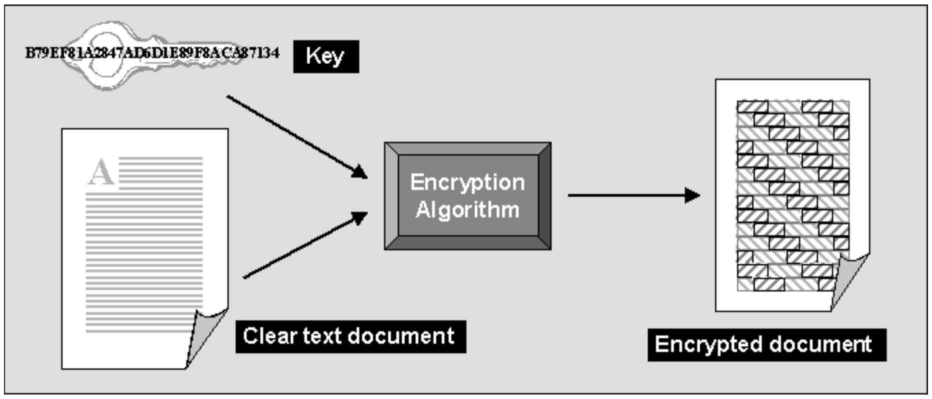


Figure 69. Encrypting a document using a key

The trading engine encryption method

The trading engine uses a combination of public-private key encryption, which is also known as asymmetric encryption, and symmetric key encryption. This hybrid system uses the best characteristics of each method and minimizes the shortcomings of each. It follows the widely adopted S/MIME standard for securing messages.

The advantage of symmetric key encryption is that it performs the encryption task more quickly than asymmetric encryption. The advantage of asymmetric encryption is that it allows you to send an encrypted message to a partner who does not hold your secret key.

To use the best of both, the trading engine uses the faster symmetric key to encrypt the document, such as a lengthy EDI transaction set, and the asymmetric key for the smaller task of encrypting the one-time session key. The session key can then be securely included with the message for transmission and allows your partner to decrypt the contents without sharing your secret key.

Symmetric key lengths

The trading engine supports several key lengths for the symmetric key you choose. You need to be careful to choose a key length your partner can support.

Public-private (asymmetric) key algorithms

The trading engine uses the RSA cryptosystem for asymmetric encryption and the digital signatures provided by using certificates.

You can use two types of asymmetric RSA keys:

- Keys issued to you, typically by a certificate authority, and subsequently imported into the trading engine. Such keys are sometimes called managed keys.
- Keys generated by you in the trading engine. Such keys are called self-signed keys.

Public-private (asymmetric) key lengths

The trading engine supports encryption key lengths of 512, 1024, and 2048 bits for the public-private key. You must choose one of these key lengths when you generate or obtain your certificate. You do not need to choose the same key length as your trading partner.

Support for dual keys

Some EDIINT-interoperable software products use two keys: one for encrypting documents and the other for signing documents. The trading engine supports single- and dual-key certificates. You do not need to do anything different to trade documents with a partner who uses dual keys.

Encryption and signing summary

Described in the simplest terms, the trading engine exchanges encrypted and signed documents in S/MIME format.

Outbound documents

The document contains the data that needs to be protected. The encryption and signing processes take place for every document that the trading engine sends over the Internet.

The trading engine encrypts and signs each document by building three parts: the encrypted document, the encrypted session key and the digital signature. The following is the process for an outbound document.

1. A hashing routine (MD5 or SHA-1) creates a digital digest of the document. This digest is a number. If the data in the transaction are changed, added to or subtracted from, reapplying the hashing routine will produce an entirely different digest. This characteristic of hashing routines makes it easy for a partner to verify the integrity of an inbound document.

2. The digital digest is encrypted using your private key. This encrypted digest is the digital signature for this document. It ensures that the data in the document were not changed and that the document came from you and only you.

3. The trading engine generates a one-time session key. This is the symmetric key part of the trading engine’s hybrid encryption method.

4. The session key is used to encrypt the document.

5 Your partner's public key is provided in the certificate inside the profile your partner gave you. It is used to encrypt the session key for transmission. Thus, the key to decrypting the document has itself been encrypted by your partner's public key and can be decrypted only by your partner's private key.

6 The document is then sent using whatever transport method you choose for this partner.

Inbound documents

When a document is received by your trading partner, the process is reversed according to the following steps.

1 Upon receiving the document, the trading engine begins security processing.

2 Your partner uses the private key (the matching half to the asymmetric public key you used to encrypt it) to decrypt your symmetric key.

3 The one-time key that was just decrypted is used, in turn, to decrypt the document. Your partner now has your message in clear text.

4 With the public half of your public-private key pair that you sent your trading partner in your certificate (inside your community profile), your trading partner decrypts the digital signature.

5 Your partner uses the same hashing routine (MD5 or SHA-1) to create a digital digest of the document. This is called rehashing. Your trading partner then compares this to the digest in the digital signature you sent. If the two are identical, your partner has proof that the contents of the document were not altered and that it came from you and only you.

6 The document is now ready to be read into and used by your partner's business application.

Any documents that cannot be successfully processed are failed.

Ensuring data integrity and trust

When digitally signed, ensuring data has not changed and can be trusted involves two steps:

1 Verifying the signature

2. Validating the verification certificate

The verification certificate is the certificate containing the public key corresponding to the private key that was used to create the signature in the first place. This certificate is almost always provided as part of the signature that is transported along with the signed data.

Signature verification

Signature verification consists of the following steps.

1. Compute a hash value over the signed data.
2. Using the public key in the verification certificate, decrypt the encrypted hash value in the signature.
3. Ensure the two hash values are equal. If so, the signature is verified. It is known the data has not been changed since it was signed.

Certificate path validation

Certificate path validation ensures a public-key certificate has not been tampered with and can be trusted. All certificates are signed by their issuing certificates. This means each certificate contains a signature that can be checked through the signature verification process previously described. The verification ensures the certificate has not been tampered with. For a given end-entity certificate, the list of certificates from itself through its intermediate certificates to its root certificate is known as the certificate path or chain. (Self-signed or root certificates are signed by themselves.)

Validating a certificate consists of the following steps.

1. Construct the path from the certificate to its root certificate.
2. Verify the signature of each certificate in the path.
3. Ensure that each certificate in the path has not expired.
4. Ensure that each certificate in the path has not been revoked. See Certificate revocation lists on page 359.
5. Ensure at least one certificate in the path is trusted. A certificate is trusted if it appears in the appropriate trusted root store (also know as a PSE or personal security environment).

Cyclone Activator must always be able to build and validate the complete path of certificates from verification certificate to its root certificate. However, under security implemented for some other systems, the process stops with the first encounter of a trusted certificate.

Certificate basics

A certificate contains the public half of your public-private key pair along with other identifying information about your community profile and point of contact. You use the public key in your partner's certificate to encrypt a document for transmission over the Internet. Your partner uses the public key in your certificate to verify the digital signature of a document received from you.

The following is some basic information about how the trading engine uses certificates:

- A community must have a certificate to exchange secure documents. The trading engine can generate the certificate or it can be generated externally.
- Each partner also must have a certificate as part of the partner profile.
- A community or partner profile can have only one certificate designated as the default encryption or signing certificate.
- A community or partner profile can have multiple certificates.
- The key length for a community certificate does not have to be the same as for a partner's certificate.

SSL authentication

The trading engine optionally allows certificates to be used for authenticating the identity of trading partners. Secure Sockets Layer (SSL) protocol authentication provides an added layer of security to trading relationships.

A community can be in the client or the server role when trading with a given partner.

- **Community in client role**. When the community is sending a message to a partner, it acts as the SSL client. This requires the community to have trusted the partner's SSL server certificate.

- **Community in server role**. When the partner is sending a message to the community's embedded SSL server, the community acts as the SSL server. If the embedded SSL server has been configured to require client authentication, the community must have trusted the partner's SSL client certificate.

At the time of setting up an outbound delivery exchange, you specify that "clients must use SSL to connect to this server." You can further specify to "enable host name verification." The first control requires use of SSL protocol during connections. The second optional control makes the trading engine compare the name of the SSL server to the name in the server's certificate to make sure they are the same.

When setting up an inbound delivery exchange, you also can specify that "clients must use SSL to connect to this server." Optionally, you also can require "client-side certificate authentication," which means a partner's certificate is used to verify the partner's identity when a connection is made.

These controls are further described in Delivery exchanges on page 181 and Transport maintenance on page 247.

Note: If you have a partner who uses webMethods, and the webMethods server runs HTTPS and requires client authentication, and you have not selected an SSL client authentication certificate, the connection will be closed. The reason will not be apparent in Cyclone Activator. Cyclone Activator will produce a socket closed error message, but not indicate the SSL handshake failed. To resolve this, select a certificate for SSL authentication in the community profile.

The following summarizes what happens when a client connects to an SSL server. These steps apply whether the community is connecting to the partner's SSL server (meaning the community is playing the client role) or the partner is connecting to the community's SSL server (meaning the community is playing the server role). Note that the way the trading engine performs these tasks may not precisely mirror this order. The steps are presented to illustrate the various checks that may occur.

1. The client establishes a socket connection to the SSL server. This could be an HTTP, FTP or another kind of server.
2. The server sends the client its SSL server certificate. This is a required step in the SSL handshaking sequence.
3. The client checks whether it trusts the server's certificate.

If the client trusts the server's certificate, the connection is maintained. Otherwise, the client drops the connection if it does not trust the server's certificate.

This is the end of the authentication process, unless the server is configured to require client authentication. If client authentication is called for, the following additional steps are performed.

4. The server explicitly asks the client to send its SSL client certificate.
5. The client sends the server its SSL client certificate.
6. The server checks whether it trusts the client's certificate.
7. If the server trusts the client's certificate, the authentication process is completed (unless host name verification is required as noted in the next step). Otherwise, the server drops the connection if it does not trust the client's certificate.
8. If host name verification also is called for, the client takes the additional step of comparing the name of the SSL server to the name in the server's certificate. If the names are not the same, the client drops the connection.

Giving certificates to partners

Before you can exchange encrypted and signed documents with a trading partner, each of you must obtain the other's certificate and public key. You do this after you have created your community profile. Each of you obtains a certificate with a public-private key pair, either by generating a self-signed certificate or obtaining a certificate from a certificate authority.

The private half of your key pair always remains on your computer. The public half is given to your partners when you send them your community profile, which includes the certificate and public key, or the certificate alone.

The following topics describe how to give your certificate to partners. In all cases, we recommend confirming the certificate fingerprints with your trading partner before exchanging documents.

Certificate exchange with Cyclone partners

If your trading partner also uses Cyclone Commerce software, export your community profile, which includes your certificate and public key, to a file and send that file to the partner. We recommend doing so by a secure means.

Certificate exchange with other partners

If you exchange encrypted and signed documents with partners who use other interoperable software, export your certificate and public key to a file and send the file to partners. For initial distribution of self-signed certificates, we recommend sending the certificate to each partner on diskette by a secure means. Subsequent distribution can be on diskette or by e-mail. For more information see Export a certificate to a file on page 355.

Self-signed or CA certificates

You and your partners should decide whether to use self-signed certificates or certificates from a third-party certificate authority. Consider the following when deciding:

- Self-signed certificates are easily created. The primary disadvantage is lack of verification by a trusted third party.
- The primary advantage of CA certificates is the identity of the certificate holder is verified by a trusted third party. Disadvantages include the extra cost and administrative effort.
- A CA provides a centralized source for posting and obtaining information about certificates, including information about revoked certificates.

When to get certificates

You can generate or obtain new certificates when:

- You know or suspect a certificate has been compromised.
- You need to replace a certificate that is about to expire.
- You want to change your encryption key at planned intervals just as you would change a password.

For procedure see Set up certificates for a community on page 335 or Import certificates for partners on page 352.

If possible, perform certificate updates when your server is not processing outbound documents. By observing this precaution you can avoid documents being rejected by partners.

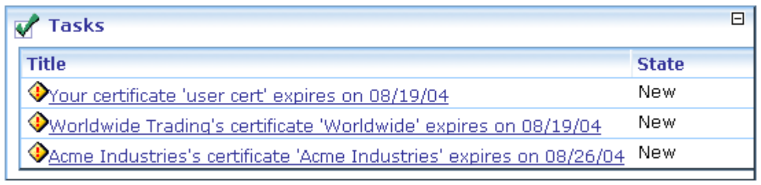
If you generate a new self-signed certificate because the old one has expired, become defective or corrupted or cannot be used for any other reason, you should export your certificate to a file and distribute it to your partners on diskette by a secure means, which includes in-person, U.S. mail or private delivery service.

What to do with expiring certificates

Using certificates to ensure security in document exchanges between partners is optional, but preferred. When sending a message, the partner's public key in a certificate file is used to encrypt the message. If the certificate is expired, the trading engine will not encrypt or send the message. Likewise, an inbound encrypted message cannot be deciphered with an expired certificate. It is important to make sure the certificates associated with community and partner profiles are current and have not passed their expiration dates.

The trading engine posts a message on the user interface home page 14 days before a community or partner certificate expires. It also displays an alert message on the Alerts toolbar menu. If your license allows users to have certificates (for example, CSOS functionality), the trading engine also generates messages about user certificates that are about to expire.

The messages about expiring certificates remain until the certificates are deleted. Figure 70 shows messages on the home page about expiring certificates.



Title	State
Your certificate 'user cert' expires on 08/19/04	New
Worldwide Trading's certificate 'Worldwide' expires on 08/19/04	New
Acme Industries's certificate 'Acme Industries' expires on 08/26/04	New

Figure 70. Messages on home page warning of certificates about to expire

The messages give you time to replace certificates before they expire. We recommend replacing certificates before rather than after expiration so trading is not disrupted. Regardless, expired certificates must be replaced. They cannot be used for encryption, decryption or signing.

Do the following when a certificate is about to expire. The advice about archiving expired certificates is recommended, but not required.

1. If a partner's certificate is about to expire, notify the partner and ask for a replacement.
2. In [install directory]\common create a subdirectory named **certarchive**. Create subdirectories of certarchive named **community** and **partner**.
3. On the home page click the message about an expiring certificate to open the certificate's maintenance page.
4. Click **Export this certificate**.

   If a community or user certificate, select the option to export the private key to a .p12 file. Save the file in [install directory]\common\certarchive\community.

   If a partner certificate, select the option to export the public key to a .p7b file. Select **Include all certificates in the certificate path if possible**. Save the file in [install directory]\common\certarchive\partner.
5. Obtain a replacement certificate.

   If a community certificate, create a self-signed certificate or obtain a CA certificate. See Set up certificates for a community on page 335

   If a user certificate, see CSOS orders on page 503.

   If a partner certificate, import the replacement certificate the partner sends you. See Import certificates for partners on page 352.
6. Delete the old certificate. On the community or partner summary page, click **Certificates** on the navigation graphic at the top of the page, select the certificate and click **Delete this certificate**. If a user certificate, open the user maintenance page certificates tab, select the certificate and click **Delete this certificate**.

Trusted roots

Trusted roots are the foundation upon which chains of trust are built in CA certificates. Underlying a certificate issued by a certificate authority is a root, self-signed certificate. There also can be intermediate certificates in the chain. In the trading engine trusting a CA root means you trust all certificates issued by that CA. Conversely, if you elect not to trust a CA root, the trading engine will not trust any certificates issued by that CA. Document trading fails in the trading engine when a non-trusted certificate is used.

The self-signed certificates you can generate in the trading engine are root certificates. This is because you are, in effect, your own CA when you generate a self-signed certificate.

The trading engine by default trusts your self-signed certificates that it generated. The trading engine also by default trusts the roots of the CA-issued certificates of a community's partners.

The trusted root certificates tab in the user interface displays all of the root certificates that your community trusts, including those of certificate authorities.

Cyclone Activator is pre-loaded with intermediary and trusted root certificates at [install directory]\[build number]\conf\certs. The pre-loaded roots are not trusted, but are simply available in the certificate store for validating end-entity certificates as they are imported and used.

Importing a trusted root is a task that rarely, if ever, must be performed. You might have to import a trusted root if, for example, your partner sends you a CA-issued certificate and your system does not have the trusted root for it. In such a case, document trading would fail. As a solution, you would need to import the root underlying the certificate and trust it.

The trading engine can import trusted roots contained in files with the following extensions: .cer, .crt, .der, .p7b and .p7c. Using a directory hierarchy, as Cyclone Activator does in \conf\certs, is recommended for arranging certificates by issuer.

There are various ways you can obtain such trusted root files:

- You can use the trading engine to export a certificate file with an extension of .p7c. See Export a certificate to a file on page 355.
- You can check whether trusted root files are available for download on the web site of the public CA that issued the certificate.
- If the certificate was issued by an in-house CA such as Entrust, you can ask the CA administrator for a trusted root file.
- If the certificate is present in a browser, you can use the application's trusted roots option to export the trusted root to a file.

Trusted root certificate files can be imported one by one in the user interface. Alternately, you can copy trusted roots en masse to [install directory]\[build number]\conf\certs, where the certificates are loaded when the server is restarted. See Auto importing of intermediate and root certificates on page 323.

When you import a trusted root for a certificate to the trading engine, we recommend that you compare the MD5 fingerprints in both the trusted root and the certificate to verify that they match.

Auto importing of intermediate and root certificates

This topic is helpful when you want to automatically import intermediate and root certificates not already available in Cyclone Activator. This is an uncommon case most users will not encounter.

To successfully trade using CA-issued certificates, Cyclone Activator must be able to establish the chain of trust running through end-entity, intermediate and root certificates. This is why the trading engine is pre-loaded with many intermediate and root certificates issued by various CAs. These certificates are available for trusting upon importing end-entity certificates containing public-private encryption key pairs.

The pre-loaded intermediate and root certificates are at [install directory]\[build number]\conf\certs. Figure 71 shows part of the certs directory hierarchy on a Windows file system. Notice that certificates are organized by CA. Each CA folder has a Root subdirectory and, if needed, an Intermediate subdirectory. Certificates in these directories are loaded into the database upon starting the server the first time. If certificates are added, these are loaded into the database when the server is re-started.

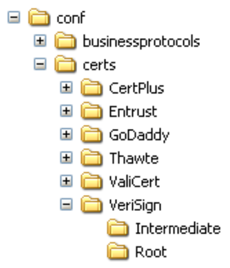


Figure 71. [install directory]\[build number]\conf\certs

If you need to add certificates, copy the files to the directory for the appropriate CA. If a CA is not already represented, add a directory for it.

Typically, root certificates have extensions of .cer, .crt or .der. Add root certificates to the Root directory for the appropriate CA. Intermediate certificates should have extensions of .p7b or .p7c. An intermediate certificates should contain both the intermediate certificate and the root certificate.

The trading engine ignores any files in the certs directory with extensions other than .cer, .crt, .der, .p7b and .p7c. So you can add readme files if you want to document added certificate files.

Errors or warnings that occur when certificates are imported are written to the server.log file.

❖ ❖ ❖

18 Manage certificates

The trading engine offers true security by providing authentication, confidentiality, integrity and non-repudiation of documents. The trading engine uses state-of-the-art cryptography to ensure the security of the documents you exchange over the public Internet.

Note: If your software license allows users to have certificates, see CSOS orders on page 503.

Procedure

- View certificate information on page 329
- Add a certificate on page 334
- Set up certificates for a community on page 335
- Import certificates for partners on page 352
- Export a certificate to a file on page 355
- Delete certificate on page 358
- Add a CRL on page 362
- Manage CRLs on page 363

Pages and fields

- Certificates pages on page 325
- Community certificates page on page 326
- Partner certificates page on page 328
- Certificate field descriptions on page 329

Concepts

- Certificate revocation lists on page 359
- Advanced CRL settings on page 364

Certificates pages

The certificates pages let you manage certificates for your community and partner profiles. Open a certificates page by clicking **Certificates** on the navigation graphic at the top of the community or partner summary page.

Using the certificates page you can:

- View a list of all certificates for a community or partner.

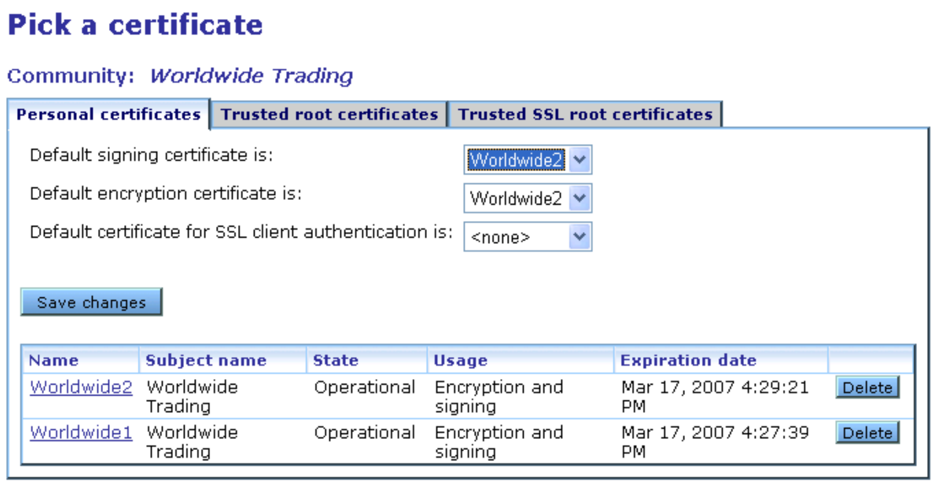
- View detailed information about certificates.

- Open the certificate wizard to generate or import a key pair and certificate for a community profile.
- Export a certificate and public key to a file for transmittal to your partners.
- Import a partner's certificate.
- Delete a certificate.
- Designate a different certificate as the default used by a community or partner.

Community and partner certificate pages

The pages for listing the certificates of communities and partners are different, but have some of the same information. You open the certificate pages the same way for both a community and a partner, by clicking **Certificates** on the navigation graphic at the top of a community or partner summary page.

The following topics explain the community and partner certificate pages.



Community: *Worldwide Trading*

Default signing certificate is:	Worldwide2
Default encryption certificate is:	Worldwide2
Default certificate for SSL client authentication is:	<none>

Name	Subject name	State	Usage	Expiration date	
Worldwide2	Worldwide Trading	Operational	Encryption and signing	Mar 17, 2007 4:29:21 PM	Delete
Worldwide1	Worldwide Trading	Operational	Encryption and signing	Mar 17, 2007 4:27:39 PM	Delete

Figure 72. Certificates page for a community

Community certificates page

On the community certificate page:

Personal certificates tab

The personal certificates tab displays the default certificates, if any, for signing documents, encrypting documents and for use in authenticating SSL connections (see SSL authentication on page 316). It also lists all certificates associated with the community along with state, usage and expiration dates.

The displayed certificates also are known as end-entity certificates. In the case of CA-issued certificates, end-entity certificates have a chain of trust that includes intermediate and root CA certificates. In the case of a self-signed certificate, it is both the end-entity and root certificate.

If the community has more than one certificate, you can select another as the default certificate and click **Save changes**.

Valid certificate states are:

Operational means the certificate is valid and can be used. This state only means the certificate can be used, not that it is in use. The trading engine rejects an outbound message when packaging is attempted with an expired or revoked certificate.

Expired means the certificate is past its validity period and no longer can be used.

Revoked means a certificate authority has invalidated the certificate and it no longer can be used.

Pending means a certificate is not yet valid, usually because it is before the date that begins the certificate's validity period.

Failed means the certificate is corrupted and cannot be used, usually due to an error while importing it to the certificate store.

Trusted roots certificates tab

The trusted roots certificates tab displays the roots of partners' certificates that a community trusts. In the case of a self-signed certificate, the trust is for the certificate itself, as a self-signed certificate is a root certificate. In the case of a certificate authority certificate, the trust is for the root certificate in the chain of trust of a partner's certificate. The system by default trusts root certificates when end-entity partner certificates are imported.

You can elect not to trust a root certificate by clicking **Untrust** to the right of the root certificate name. If you do untrust, the system no longer recognizes the end-entity partner certificate as valid.

This could affect many end-entity partner certificates. You cannot restore trust on this tab. To trust the roots again, import the partner end-entity certificate or the root certificate. The easier method is importing the end-entity certificate, as the system trusts the root by default.

A single CA might be listed multiple times on the tab, because each has multiple roots, each with unique fingerprints under which it issues certificates. To view the fingerprints, select a root and review the MD5 and SHA1 fingerprints on the details tab. By comparing fingerprints you can choose to trust some but not all of a CA's certificates. See MD5 and SHA1 fingerprints on page 333.

To import a trusted root, click **Add a trusted root certificate** on the certificates page and see Import certificates for partners on page 352.

Trusted SSL root certificates tab

The trusted SSL root certificates tab displays the trusted roots of partners' certificates that, when presented by partners, enable the partners to connect to the community's SSL servers that require client authentication. For an explanation of trusted roots and the consequences of untrusting, see Trusted roots certificates tab on page 327. Also see SSL authentication on page 316.

To import a trusted root, click **Add a trusted root certificate for SSL servers** on the certificates page and see Import certificates for partners on page 352.

Partner certificates page

The partner certificate page displays the default certificate, if any, for encrypting documents. It also lists all certificates associated with the partner along with state, usage and expiration dates.

The displayed certificates also are known as end-entity certificates. In the case of CA-issued certificates, end-entity certificates have a chain of trust that includes intermediate and root CA certificates. In the case of a self-signed certificate, it is both the end-entity and root certificate. The trusted roots of partner end-entity certificates are displayed on the trusted root certificate tabs of the communities that trust them. See Community certificates page on page 326.

If the partner has more than one certificate, you can select another as the default certificate and click **Save changes**.

Valid certificate states are:

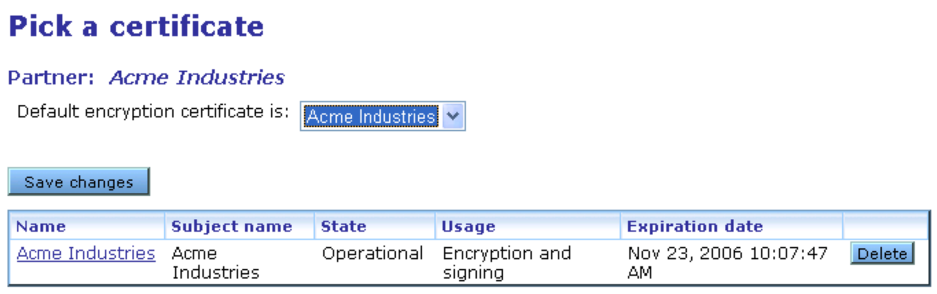
Operational means the certificate is valid and can be used. This state only means the certificate can be used, not that it is in use.

Expired means the certificate no longer can be used.

Revoked means a certificate authority has invalidated the certificate and it no longer can be used.

Pending means a certificate is not yet valid, usually because of a difference between the valid date and time in the certificate and the host clock.

Failed means the certificate is corrupted, usually due to an error while importing it to the certificate store.



Pick a certificate

Partner: *Acme Industries*

Default encryption certificate is: Acme Industries

Name	Subject name	State	Usage	Expiration date	
Acme Industries	Acme Industries	Operational	Encryption and signing	Nov 23, 2006 10:07:47 AM	Delete

Figure 73. Certificates page for a partner

View certificate information

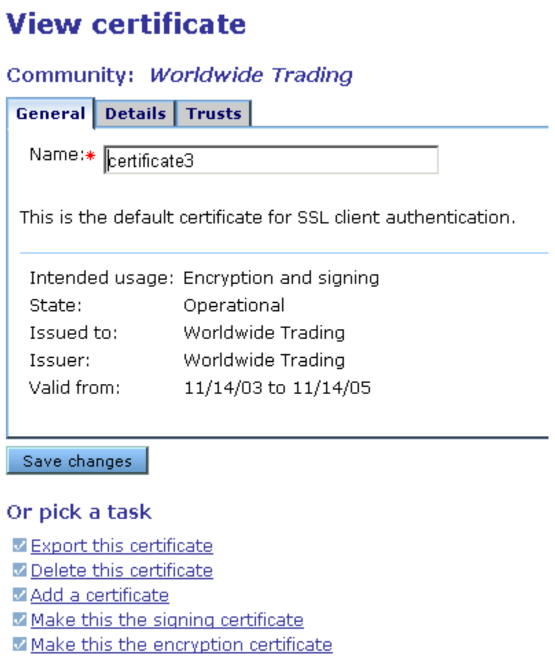
You can view information about a certificate for a community or partner. Open the certificates page by clicking **Certificates** on the navigation graphic at the top of the community or partner summary page. Click the name of a certificate to open the information page, which consists of three tabs.

If you change anything, click **Save changes**.

The following topic explains the three tabs on the certificate information page.

Certificate field descriptions

The following describes the fields on the three certificate tabs.


View certificate

Community: Worldwide Trading

Name:* certificate3

This is the default certificate for SSL client authentication.

Intended usage:	Encryption and signing
State:	Operational
Issued to:	Worldwide Trading
Issuer:	Worldwide Trading
Valid from:	11/14/03 to 11/14/05

Or pick a task

- Export this certificate
- Delete this certificate
- Add a certificate
- Make this the signing certificate
- Make this the encryption certificate

Figure 74. View certificate general tab for a self-signed certificate

General tab

Name

A user-defined name for a certificate. Naming the certificate can help identify the community or partner it belongs to.

Immediately below the Name field, one or more messages might be displayed. Such messages provide information about the certificate’s status. Possible messages are:

This is the default signing certificate. This message indicates the certificate is the default for signing documents.

This is the default encryption certificate. This message indicates the certificate is the default for encrypting documents.

This is the default SSL certificate. This message indicates the certificate is the default certificate submitted to servers to authenticate your identity. See SSL authentication on page 316.

Intended usage

Describes the functions that the certificate can perform. The intended usage does not mean the certificate is being used for that purpose, only that it can.

State

Indicates whether the certificate can be used.

Valid states are:

Operational means the certificate is valid and can be used. This state only means the certificate can be used, not that it is in use.

Expired means the certificate no longer can be used.

Revoked means a certificate authority has invalidated the certificate and it no longer can be used.

Pending means a certificate is not yet valid, usually because of a difference between the valid date and time in the certificate and the host clock.

Failed means the certificate is corrupted, usually due to an error while importing it to the certificate store.

Issued to

The name of person or entity who was issued the certificate.

Issued by

The name of the person or entity that issued the certificate. If the issued to and by names are the same, the certificate is self-signed.

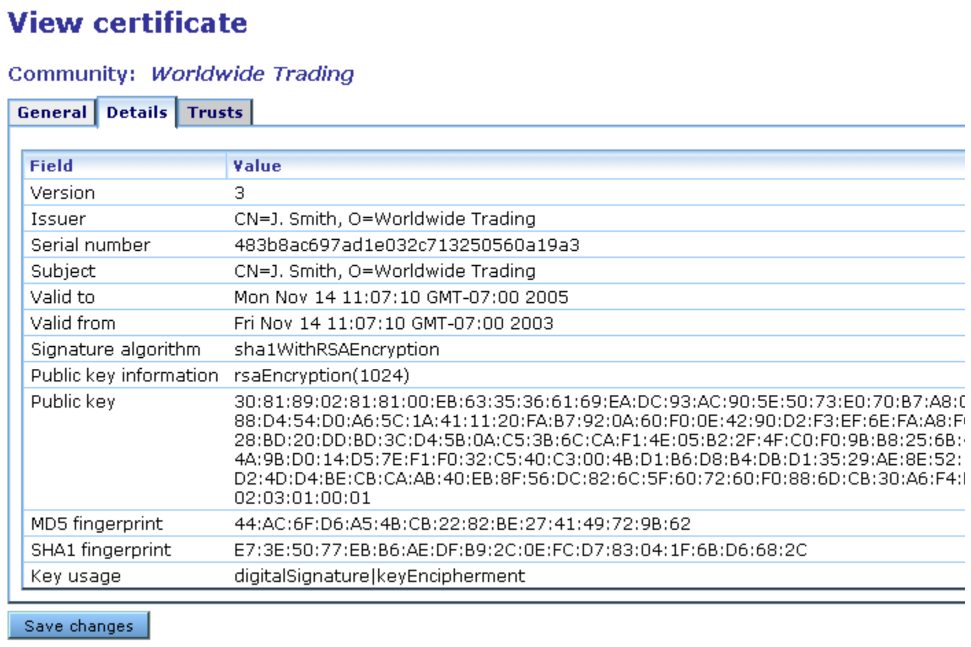
Valid from

The date range the certificate is valid.

Certificate path

If a CA certificate, the certificate path or chain of trust for the certificate appears. This field does not apply to self-signed certificates.

A chain of trust or certificate chain is an ordered list of certificates that includes the certificate of the end-user and certificates of the issuing CA. A trusted root is a public key that is verified as belonging to an issuing CA, which is called a trusted third party.



Field	Value
Version	3
Issuer	CN=J. Smith, O=Worldwide Trading
Serial number	483b8ac697ad1e032c713250560a19a3
Subject	CN=J. Smith, O=Worldwide Trading
Valid to	Mon Nov 14 11:07:10 GMT-07:00 2005
Valid from	Fri Nov 14 11:07:10 GMT-07:00 2003
Signature algorithm	sha1WithRSAEncryption
Public key information	rsaEncryption(1024)
Public key	30:81:89:02:81:81:00:EB:63:35:36:61:69:EA:DC:93:AC:90:5E:50:73:E0:70:B7:A8: 88:D4:54:D0:A6:5C:1A:41:11:20:FA:B7:92:0A:60:F0:0E:42:90:D2:F3:EF:6E:FA:A8: 28:BD:20:DD:BD:3C:D4:5B:0A:C5:3B:6C:CA:F1:4E:05:B2:2F:4F:C0:F0:9B:B8:25:6B: 4A:9B:D0:14:D5:7E:F1:F0:32:C5:40:C3:00:4B:D1:B6:D8:B4:DB:D1:35:29:AE:8E:52: D2:4D:D4:BE:CB:CA:AB:40:EB:8F:56:DC:82:6C:5F:60:72:60:F0:88:6D:CB:30:A6:F4: 02:03:01:00:01
MD5 fingerprint	44:AC:6F:D6:A5:4B:CB:22:82:BE:27:41:49:72:9B:62
SHA1 fingerprint	E7:3E:50:77:EB:B6:AE:DF:B9:2C:0E:FC:D7:83:04:1F:6B:D6:68:2C
Key usage	digitalSignature|keyEncipherment

Figure 75. Certificate details tab for a self-signed certificate

Details tab

The X.509 standard defines the information displayed on the details tab.

Version

The version of the X.509 standard that applies to the certificate.

Issuer

The issuer is the X.500 distinguished name of the CA or entity that signed the certificate. In cases of a self-signed certificate, the issuer and subject are the same. Using the certificate implies trusting the signer.

Serial number

The serial number uniquely identifies the certificate. The CA or entity that issued the certificate assigned this number. If the issuer revokes a certificate, it can place the serial number on a certificate revocation (CRL) list.

Subject

The subject is the X.500 distinguished name of the entity whose public key the certificate identifies. A distinguished name has the following parts:

C	Two-letter ISO country code
L	City or locality name
O	Organization name
OU	Organizational unit
CN	Common name of a person

Valid to

The date the certificate expires, provided it is not compromised or revoked before that date.

Valid from

The date the certificate became valid.

Signature algorithm

The algorithm the CA used to sign the certificate.

Public key information

An algorithm identifier that specifies the public key crypto system this key belongs to and any associated key parameters, such as key length.

Public key

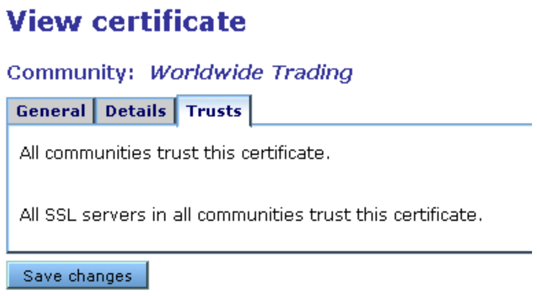
The public key of the certificate.

MD5 and SHA1 fingerprints

The fingerprints are a way to verify the source of a certificate. After you import or export a certificate, you should contact your partner and ensure that the fingerprints at both ends are identical. You should do this before you attempt to exchange documents. If the fingerprints do not match, one of the certificates might be corrupted or out of date.

Key usage

Identifies the purpose of the key in the certificate, such as encipherment, digital signature or certificate signing.


View certificate

Community: *Worldwide Trading*

All communities trust this certificate.

All SSL servers in all communities trust this certificate.

Figure 76. Certificate trusts tab for a self-signed certificate

Trusts tab

The trusts tab identifies the communities and SSL servers that do not trust the certificate.

Add a certificate

A **community** has three options for adding a certificate:

- Create a self-signed certificate. See Generate self-signed certificates on page 336
- Import a certificate and private key from a file. See Import key pair in certificate file on page 349.
- Import a certificate from a certificate authority. See the following topics:

  Import Entrust certificate on page 339
  Import RSA Keon certificate on page 343
  Import VeriSign XKMS certificate on page 346

A **partner** has the single option of importing a certificate from a file. See Import certificates for partners on page 352.

Note: If your software license allows users to have certificates, users have the same certificate options as communities. To add a certificate for a user, go to the users and roles area of the user interface, select the user you want and click **Add a certificate**. The set up procedure is the same as for a community.

Set up certificates for a community

Use this procedure to create self-signed certificates for your community profile or to load a third-party certificate for your community profile.

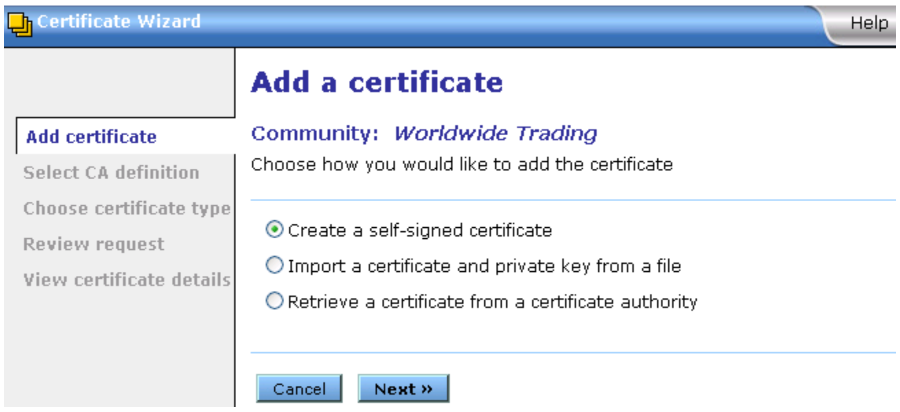
If you want to import a certificate from a third-party CA, such as VeriSign, you must obtain the certificate using your Internet browser and export it to a file before you begin this procedure. You must export the certificate to a file that contains the private key and the entire chain of trust. You will need the password used to export the file from your browser to load the certificate into the trading engine.

If your organization has a CA server, check with the server administrator about certificate generating requirements before using this procedure.

This is not the procedure to use for importing a partner's certificate. See Import certificates for partners on page 352.

Steps

1 To associate a certificate with a community, click **Certificates** on the navigation graphic on the community summary page to display the certificates page. Click **Add a certificate** to launch the certificate wizard.



Add a certificate

Community: *Worldwide Trading*

Choose how you would like to add the certificate

[✓ Create a self-signed certificate]
Import a certificate and private key from a file
Retrieve a certificate from a certificate authority

Figure 77. Certificate wizard select certificate type page

2 Select one of the following:

Create a self-signed certificate

Click if you want the trading engine to generate one self-signed certificate, for both signing and encrypting, or two self-signed certificates (dual key), one for signing and one for encrypting. Go to Generate self-signed certificates on page 336.

Import a certificate and private key from a file

Click if you want to use a third-party certificate. Go to Import key pair in certificate file on page 349.

Retrieve a certificate from a certificate authority

Select if your organization has a certificate authority server. The following certificate authorities are supported:

Entrust Technologies. Go to Import Entrust certificate on page 339.

RSA Keon. Go to Import RSA Keon certificate on page 343.

VeriSign Onsite XKMS (XML Key Management Specification). Go to Import VeriSign XKMS certificate on page 346.

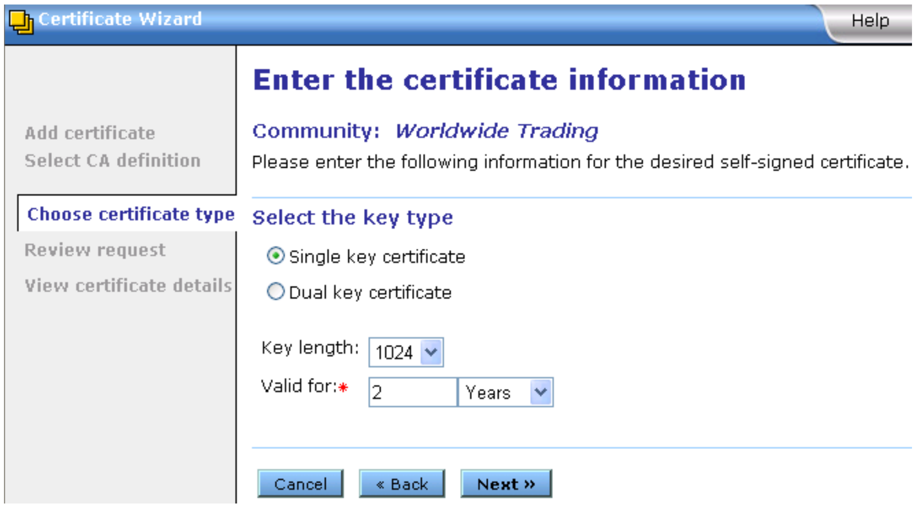
Generate self-signed certificates

Use this procedure if you selected generate self-signed certificates in step 2 of Set up certificates for a community on page 335.

The following are the steps for generating and associating with a community profile either a single self-signed certificate for both encrypting and signing documents or two self-signed certificates (dual key), one for encrypting and one for signing.

Steps

1 On the first certificate wizard page, select **Create a self-signed certificate**, and then click **Next** to display the certificate wizard select key type page.



Enter the certificate information

Community: *Worldwide Trading*

Please enter the following information for the desired self-signed certificate.

Select the key type

(●) Single key certificate

( ) Dual key certificate

Key length: 1024

Valid for:* 2 Years

Figure 78. Certificate wizard select certificate key type page

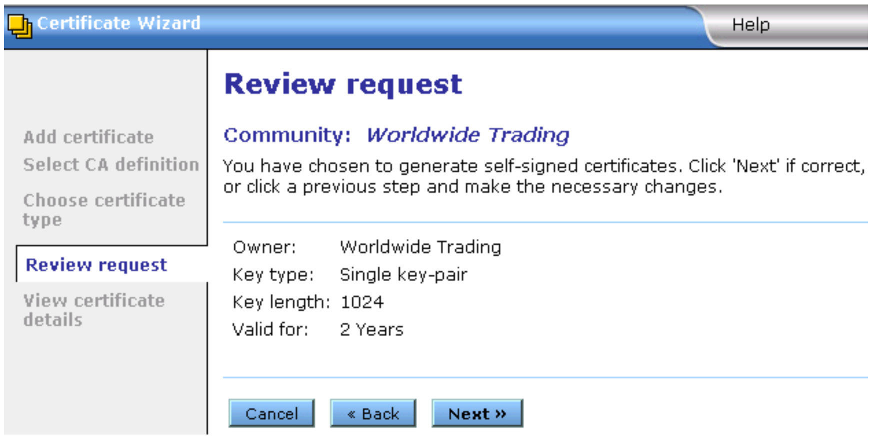
2 Click single key if you want one certificate for both signing and encrypting documents. Click dual key if you want two certificates, one for signing documents and another for encrypting documents.

3 Select one of the following encryption key lengths from the key length drop-down list:

512	Normal encryption.
1024	Strong encryption. 1024 or higher is recommended for high-value EDI transactions.
2048	Very strong encryption.

4 For the validity period, if you want other than the default value of 2 years, type the length of time you want the certificate to be valid in the validity period field. Select days, months or years from the drop-down list.

5 Click **Next** to review your certificate request.



Review request

Community: *Worldwide Trading*

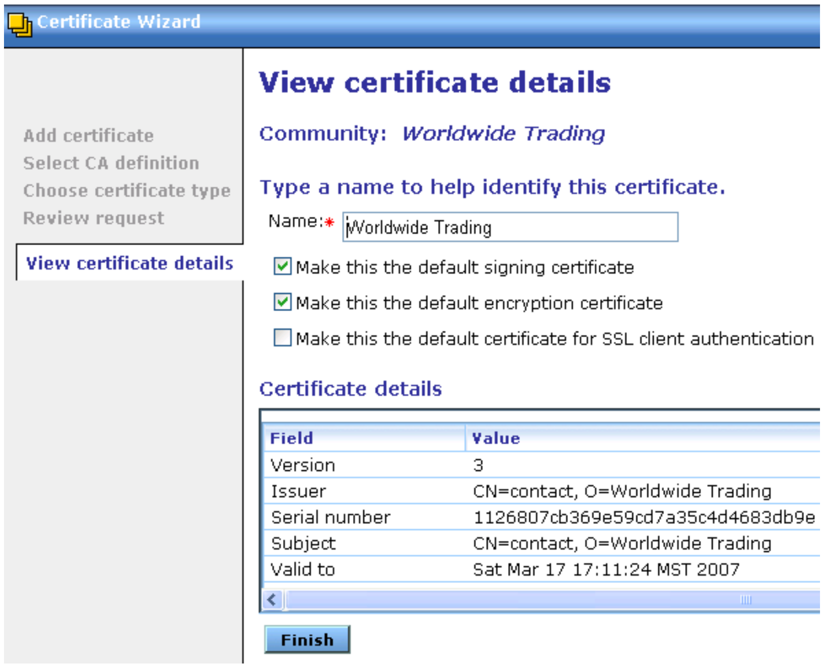
You have chosen to generate self-signed certificates. Click 'Next' if correct, or click a previous step and make the necessary changes.

Owner:	Worldwide Trading
Key type:	Single key-pair
Key length:	1024
Valid for:	2 Years

Figure 79. Certificate wizard review request page

6 Review the information on the page. If you want to make any changes. click **Back**.

7 Click **Next** to display to view the certificate details page.



View certificate details

Community: *Worldwide Trading*

Type a name to help identify this certificate.

Name:* Worldwide Trading

[✓] Make this the default signing certificate

[✓] Make this the default encryption certificate

[ ] Make this the default certificate for SSL client authentication

Certificate details

Field	Value
Version	3
Issuer	CN=contact, O=Worldwide Trading
Serial number	1126807cb369e59cd7a35c4d4683db9e
Subject	CN=contact, O=Worldwide Trading
Valid to	Sat Mar 17 17:11:24 MST 2007

Figure 80. Certificate wizard details page

8 If you want, type a name for the certificate in the Name field. This name can help you tell one certificate from another. By default the system uses the community name as the certificate name.

To make the certificate your default signing certificate, click **Make this the default signing certificate**. This option is selected by default.

To make the certificate your default encryption certificate, click **Make this the default encryption certificate**. This option is selected by default.

To make the certificate your default SSL authentication certificate, click **Make this the default certificate for SSL client authentication**. This means the community will present this certificate to a partner who requests client authentication to connect to a SSL server. See SSL authentication on page 316.

9 Click **Finish** to generate the certificate.

After the certificate is generated, the certificates page reappears and displays the new certificate.

10 If you are setting up a community profile for the first time, you must distribute your certificate information by sending it to partners on diskette or by some secure means. This can be done by exporting your certificate as part of your community profile. See Export a community profile on page 151.

If you need to distribute your certificate to your trading partners who use other interoperable software, see Export a certificate to a file on page 355.

Before you attempt to exchange encrypted and signed documents, you should contact each partner with whom you exchanged certificates and confirm that the fingerprints in both your certificates are identical. For more information see MD5 and SHA1 fingerprints on page 333.

Import Entrust certificate

Use this procedure if you selected acquire Entrust certificates in step 2 of Set up certificates for a community on page 335.

The following are the steps for importing a new Entrust certificate into the trading engine. Before you can use this procedure, you must consult with your organization’s Entrust administrator about the information required to connect with the Entrust/PKI server and import a new or updated certificate for your community profile.

The trading engine fulfills a client role in supporting the certificate management tasks of an Entrust server. The prerequisites for this client-server relationship are your Entrust server and a person who is designated as your organization's Entrust administrator. Lacking these two requirements, your organization cannot use Entrust certificates in exchanging documents with your trading partners through the trading engine.

The trading engine enables an organization with an Entrust/PKI server to create Entrust X.509 certificates.

The trading engine does not support Entrust certificate revocation or recovery.

The trading engine supports Entrust version 5.

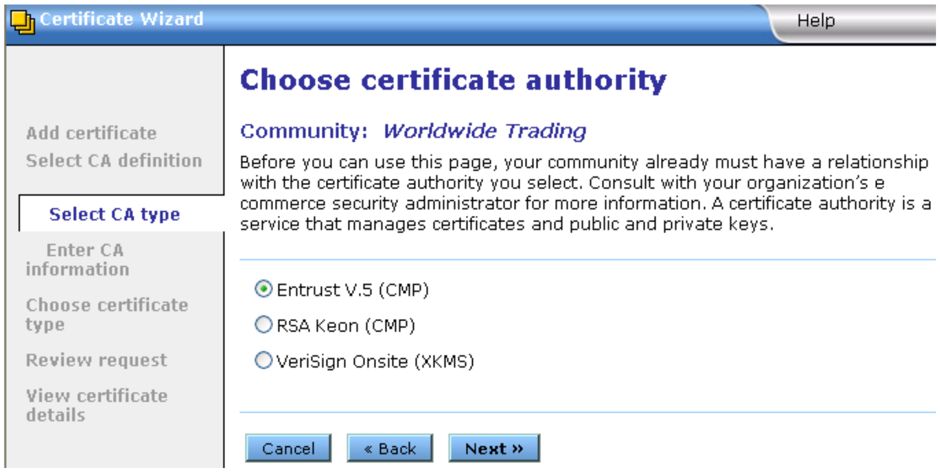
The following describes the certificate-generation process involving the trading engine and the Entrust server.

After the trading engine creates the key pair for signing documents, the application hands the public key to the Entrust server. The Entrust server creates the signing certificate and passes the certificate to the trading engine. The public key is within the certificate. The trading engine retains the private signing key. The private signing key is not disclosed to the Entrust server; the private key remains secure within the trading engine. This guarantees security integrity.

Meanwhile, the Entrust server creates the encryption key pair and creates an encryption certificate, which includes the public key. The Entrust server passes to the trading engine the encryption key pair and the encryption certificate.

Steps

1. On the first certificate wizard page, select **Retrieve a certificate from a certificate authority**, and then click **Next** to display the certificate authority selection page.
2. Select **Entrust V.5 (CMP)**.



Choose certificate authority

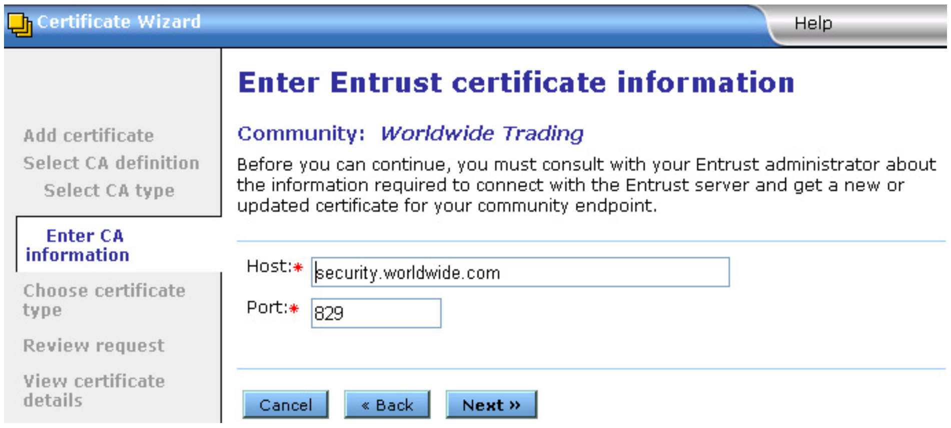
Community: *Worldwide Trading*

Before you can use this page, your community already must have a relationship with the certificate authority you select. Consult with your organization's e commerce security administrator for more information. A certificate authority is a service that manages certificates and public and private keys.

[✓ Entrust V.5 (CMP)]
RSA Keon (CMP)
VeriSign Onsite (XKMS)

Figure 81. Certificate wizard certificate authority selection page

3 Click **Next** to display the certificate wizard Entrust server information page.



Enter Entrust certificate information

Community: *Worldwide Trading*

Before you can continue, you must consult with your Entrust administrator about the information required to connect with the Entrust server and get a new or updated certificate for your community endpoint.

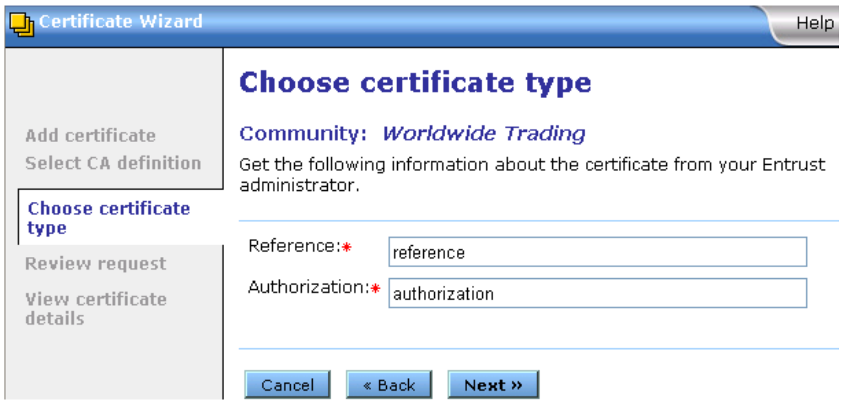
Host:* security.worldwide.com

Port:* 829

Figure 82. Certificate wizard Entrust server information page

4 Complete the host and port fields for importing the certificate. Consult with your organization's Entrust administrator to obtain the information.

5 Click **Next** to display the Entrust reference and authorization page.



Choose certificate type

Community: *Worldwide Trading*

Get the following information about the certificate from your Entrust administrator.

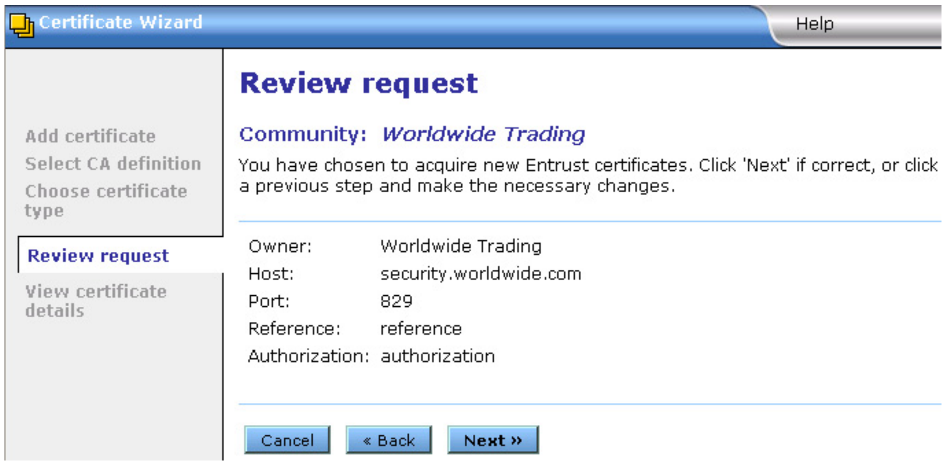
Reference:* reference

Authorization:* authorization

Figure 83. Certificate wizard Entrust reference and authorization page

6 Complete the reference and authorization fields for importing the certificate. Consult with your organization's Entrust administrator to obtain the information.

7 Click **Next** to display the certificate review request page.



Review request

Community: *Worldwide Trading*

You have chosen to acquire new Entrust certificates. Click 'Next' if correct, or click a previous step and make the necessary changes.

Owner:	Worldwide Trading
Host:	security.worldwide.com
Port:	829
Reference:	reference
Authorization:	authorization

Figure 84. Certificate wizard review request page

8 Review the information on the page. Click **Back** to change any information or click **Next** to acquire or update a certificate.

9 Click **Finish.** The certificates page reappears, displaying the new certificate.

10 If you are setting up a community profile for the first time, you must distribute your certificate information by sending it to partners on diskette or by some secure means. This can be done by exporting your certificate as part of your community profile. See Export a community profile on page 151.

If you need to distribute your certificate to your trading partners who use other interoperable software, see Export a certificate to a file on page 355.

Before you attempt to exchange encrypted and signed documents, you should contact each partner with whom you exchanged certificates and confirm that the fingerprints in both your certificates are identical. For more information see MD5 and SHA1 fingerprints on page 333.

Import RSA Keon certificate

Use this procedure if you selected acquire an RSA Keon certificate in step 2 of Set up certificates for a community on page 335.

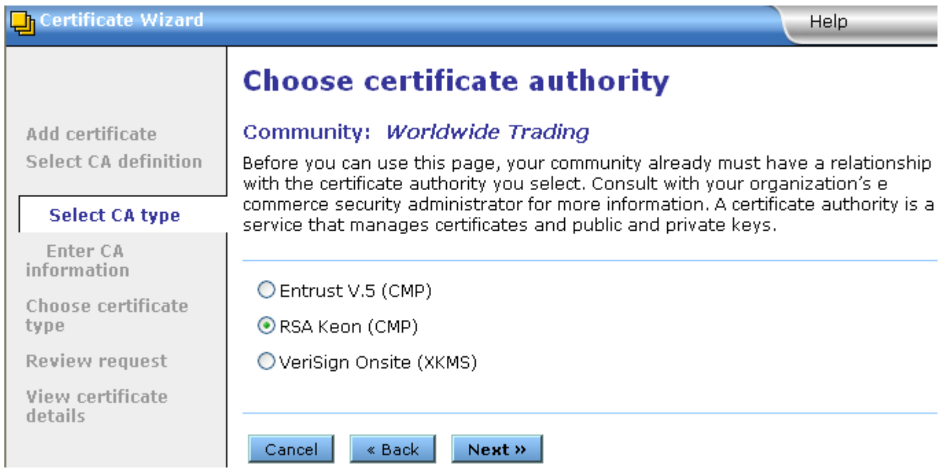
The following are the steps for importing an RSA Keon certificate into the trading engine and associating it with a community profile. Before you can use this procedure, you must consult with your organization's RSA Keon Certificate Authority administrator about the information required to connect with the Certificate Management Protocol (CMP) server and import a certificate for your community profile.

The CMP server must be running for the trading engine to acquire a certificate. Further, the RSA Keon Certificate Authority system must be configured for automatic vetting of CMP requests. For details see the certificate enrollment protocols chapter in the RSA Keon Certificate Authority user documentation.

In this process the trading engine generates the private-public key pair. The RSA Keon Certificate Authority system creates the certificate and certifies your organization as the owner of the public key.

Steps

1. On the first certificate wizard page, select **Retrieve a certificate from a certificate authority**, and then click **Next** to display the certificate authority selection page.



Certificate Wizard

Help

Add certificate
Select CA definition
Select CA type
Enter CA information
Choose certificate type
Review request
View certificate details

Choose certificate authority

Community: *Worldwide Trading*

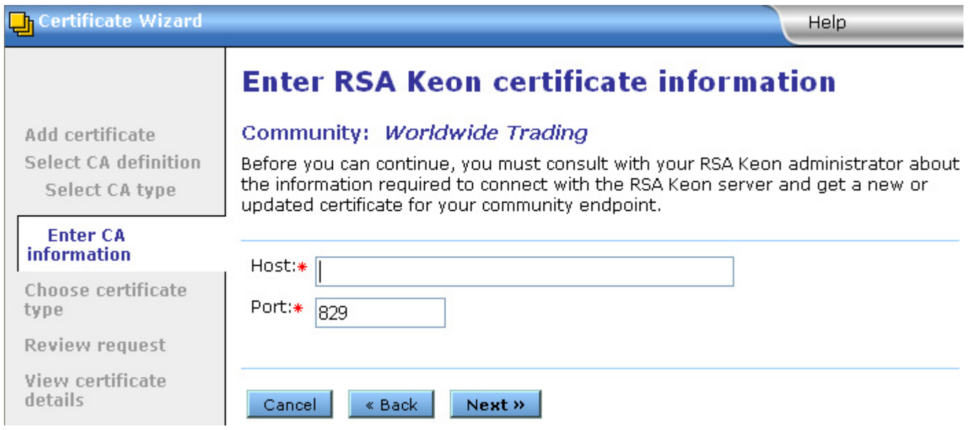
Before you can use this page, your community already must have a relationship with the certificate authority you select. Consult with your organization's e commerce security administrator for more information. A certificate authority is a service that manages certificates and public and private keys.

○ Entrust V.5 (CMP)
● RSA Keon (CMP)
○ VeriSign Onsite (XKMS)

Cancel | « Back | Next »

Figure 85. Certificate wizard select certificate authority page

2 Select **RSA KEON (CMP)** and click **Next** to display the RSA Keon host and port page.



Certificate Wizard

Help

Add certificate
Select CA definition
Select CA type
Enter CA information
Choose certificate type
Review request
View certificate details

Enter RSA Keon certificate information

Community: *Worldwide Trading*

Before you can continue, you must consult with your RSA Keon administrator about the information required to connect with the RSA Keon server and get a new or updated certificate for your community endpoint.

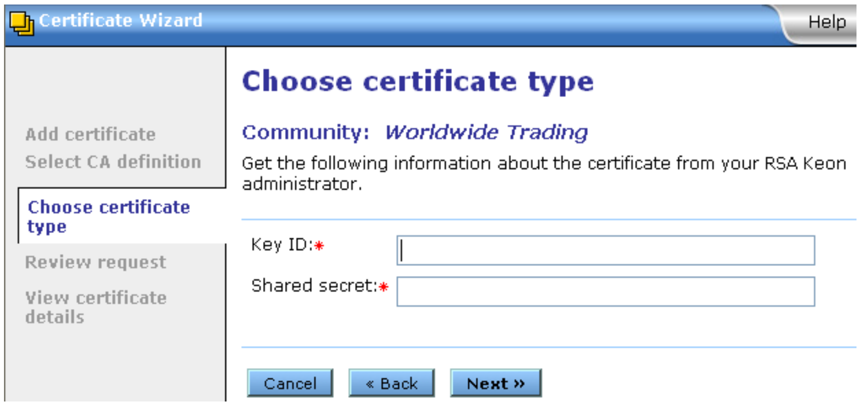
Host:*
Port:* 829

Cancel | « Back | Next »

Figure 86. Certificate wizard RSA Keon server host and port page

3 Complete the host and port fields for importing the certificate. Consult with your organization's RSA Keon administrator to obtain the information.

4 Click **Next** to display the key ID and shared secret page.



Choose certificate type

Community: *Worldwide Trading*

Get the following information about the certificate from your RSA Keon administrator.

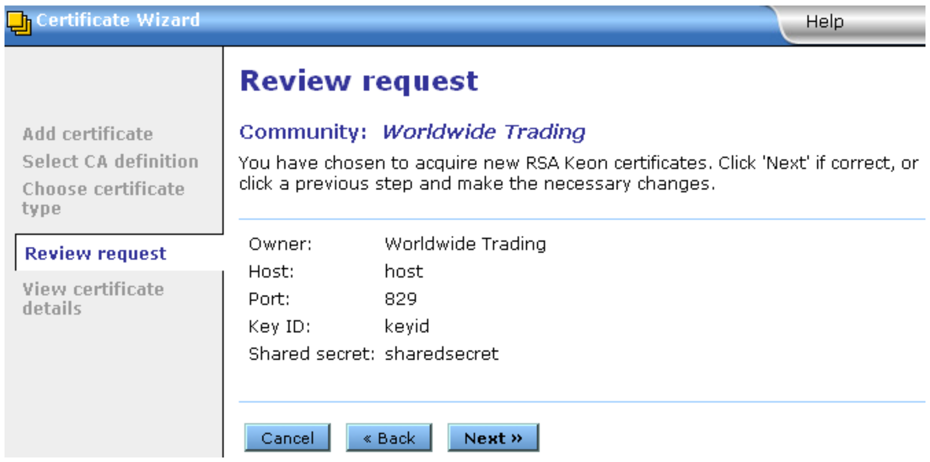
Key ID:*

Shared secret:*

Figure 87. Certificate wizard RSA Keon key ID and shared secret page

5 Using the information provided to you, complete the fields for importing the certificate. Type this information in the key ID and shared secret fields.

6 Click **Next** to display the certificate review request page.



Review request

Community: *Worldwide Trading*

You have chosen to acquire new RSA Keon certificates. Click 'Next' if correct, or click a previous step and make the necessary changes.

Owner:	Worldwide Trading
Host:	host
Port:	829
Key ID:	keyid
Shared secret:	sharedsecret

Figure 88. Certificate wizard review request page

7 Review the information on the page. Click **Back** to change any information or click **Next** to import the certificate.

8 Click **Finish.** The certificates page reappears, displaying the new certificate.

9 If you are setting up a community profile for the first time, you must distribute your certificate information by sending it to partners on diskette or by some secure means. This can be done by exporting your certificate as part of your community profile. See Export a community profile on page 151.

If you need to distribute your certificate to your trading partners who use other interoperable software, see Export a certificate to a file on page 355.

Before you attempt to exchange encrypted and signed documents, you should contact each partner with whom you exchanged certificates and confirm that the fingerprints in both your certificates are identical. For more information see MD5 and SHA1 fingerprints on page 333.

Import VeriSign XKMS certificate

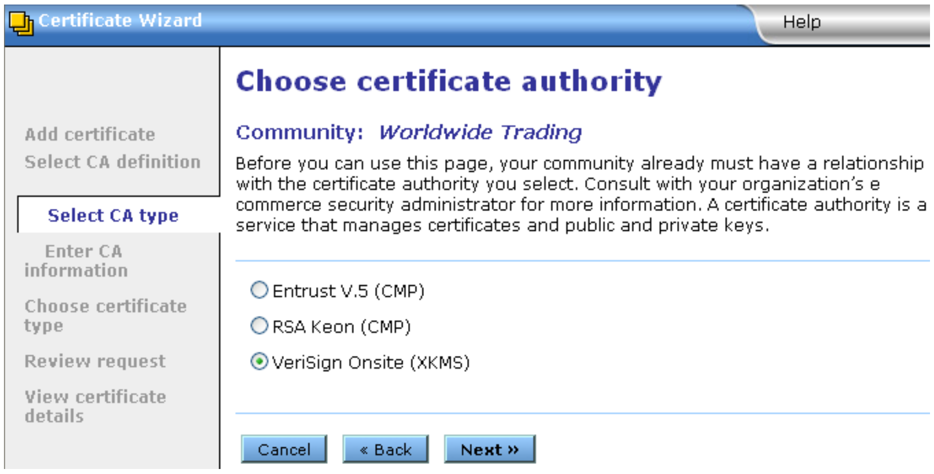
Use this procedure if you selected Retrieve a VeriSign XKMS certificate in step 2 of Set up certificates for a community on page 335.

The following are the steps for importing a new XML Key Management Specification (XKMS) certificate into the trading engine and associating it with a community profile. Before you can use this procedure, you must register for a new XKMS certificate from VeriSign. When the new certificate is ready, you will receive an e-mail containing the information needed to connect to a server and import the certificate for your community profile.

XKMS was designed in an effort to combine the interoperability afforded by Extensible Markup Language (XML) in business-to-business electronic commerce with secure and easy to use public key infrastructure (PKI). For information about XKMS see http://xmltrustcenter.org/index.htm.

Steps

1. On the first certificate wizard page, select **Retrieve a certificate from a certificate authority** and click **Next** to display the certificate authority selection page.



Choose certificate authority

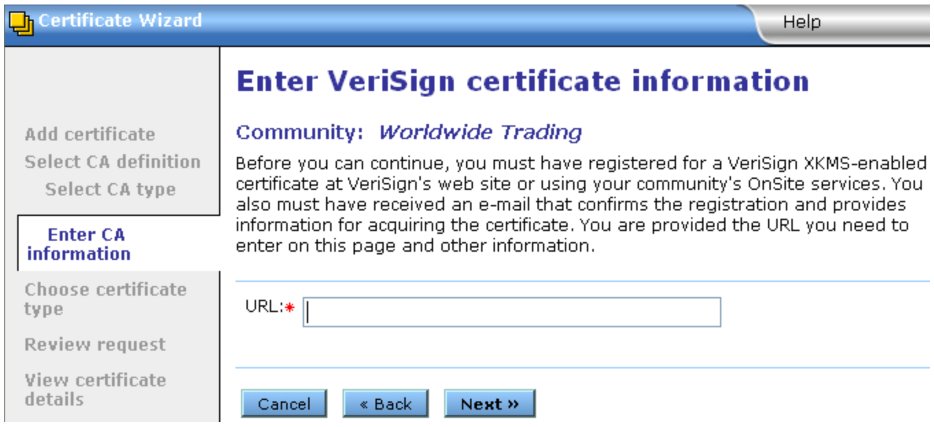
Community: *Worldwide Trading*

Before you can use this page, your community already must have a relationship with the certificate authority you select. Consult with your organization's e commerce security administrator for more information. A certificate authority is a service that manages certificates and public and private keys.

- ○ Entrust V.5 (CMP)
- ○ RSA Keon (CMP)
- ● VeriSign Onsite (XKMS)

Figure 89. Certificate wizard certificate authority selection page

2 Select **VeriSign Onsite (XKMS)** and click **Next** to display the certificate wizard VeriSign URL page.



Enter VeriSign certificate information

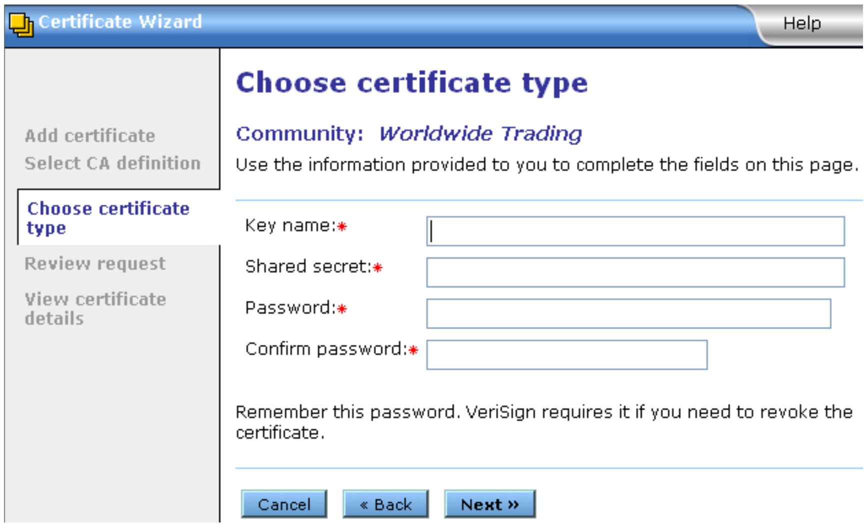
Community: *Worldwide Trading*

Before you can continue, you must have registered for a VeriSign XKMS-enabled certificate at VeriSign's web site or using your community's OnSite services. You also must have received an e-mail that confirms the registration and provides information for acquiring the certificate. You are provided the URL you need to enter on this page and other information.

URL:*

Figure 90. Certificate wizard VeriSign Onsite (XKMS) URL page

3 Using the information provided to you, complete the URL for importing the certificate and click **Next** to display the VeriSign certificate information page.



Choose certificate type

Community: *Worldwide Trading*

Use the information provided to you to complete the fields on this page.

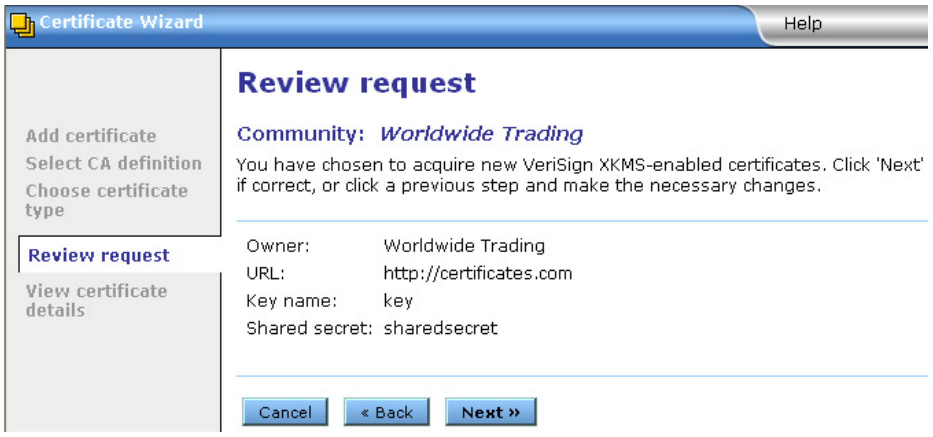
Key name:*	
Shared secret:*	
Password:*	
Confirm password:*	

Remember this password. VeriSign requires it if you need to revoke the certificate.

Figure 91. Certificate wizard VeriSign certificate information page

4 Using the information provided to you, complete the fields for importing the certificate. Type this information in the key name and shared secret fields. In the password field, type a password that you can remember. You will need this password if you later ask VeriSign to revoke the certificate.

5 Click **Next** to display the certificate review request page.



Review request

Community: *Worldwide Trading*

You have chosen to acquire new VeriSign XKMS-enabled certificates. Click 'Next' if correct, or click a previous step and make the necessary changes.

Owner:	Worldwide Trading
URL:	http://certificates.com
Key name:	key
Shared secret:	sharedsecret

Figure 92. Certificate wizard review request page

6 Review the information on the page. Click **Back** to change any information or click **Next** to import the certificate.

7 Click **Finish.** The certificates page reappears, displaying the new certificate.

8 If you are setting up a community profile for the first time, you must distribute your certificate information by sending it to partners on diskette or by some secure means.

If you need to distribute your certificate to your trading partners who use other interoperable software, see Export a certificate to a file on page 355.

Before you attempt to exchange encrypted and signed documents, you should contact each partner with whom you exchanged certificates and confirm that the fingerprints in both your certificates are identical. For more information see MD5 and SHA1 fingerprints on page 333.

Import key pair in certificate file

Use this procedure if you selected to import a certificate and private key from a file in step 2 of Set up certificates for a community.

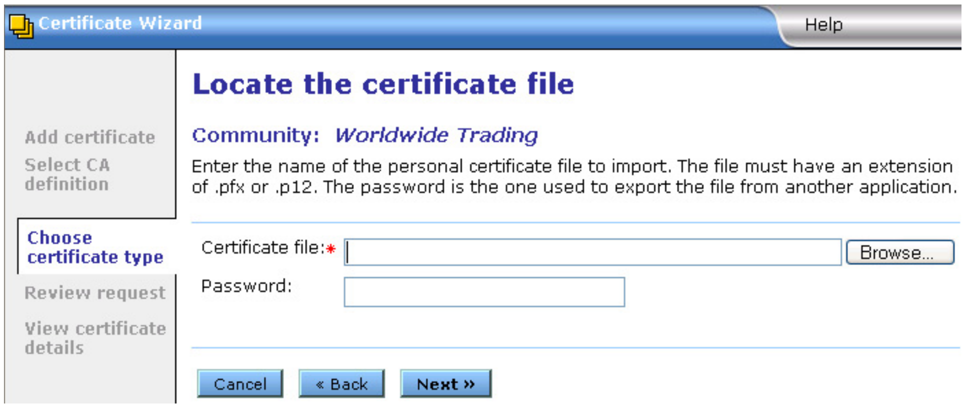
The following are the steps for importing a third-party CA certificate into the trading engine and associating it with a community profile. Such a certificate file contains both the public and private keys. Before you can use this procedure, you must perform the following tasks:

- Obtain a certificate from a certificate authority such as VeriSign.
- Export the certificate from a browser or mail client to a file. Assign a password when exporting the file; you will need this same password upon importing the file.
- Export both the public and private keys with the certificate. A certificate file with both keys is a P12 or PFX file.
- If you export the certificate from Microsoft Outlook or Internet Explorer, select the check box for "include all certificates in the certification path if possible." You want the exported file to include the entire chain of trust.

If the trading engine cannot import a P12 certificate file, import the file in Internet Explorer, making sure to mark the private key as exportable when you do so. When you have imported the certificate, view the certification path to verify that the entire path is present. Export the certificate with the private key and include all certificates in the certification path. Then try again to import the P12 file in the trading engine.

Steps

1 On the first certificate wizard page, select **Import a certificate and private key from a file** and click **Next** to display the locate the certificate file page.



Locate the certificate file

Community: *Worldwide Trading*

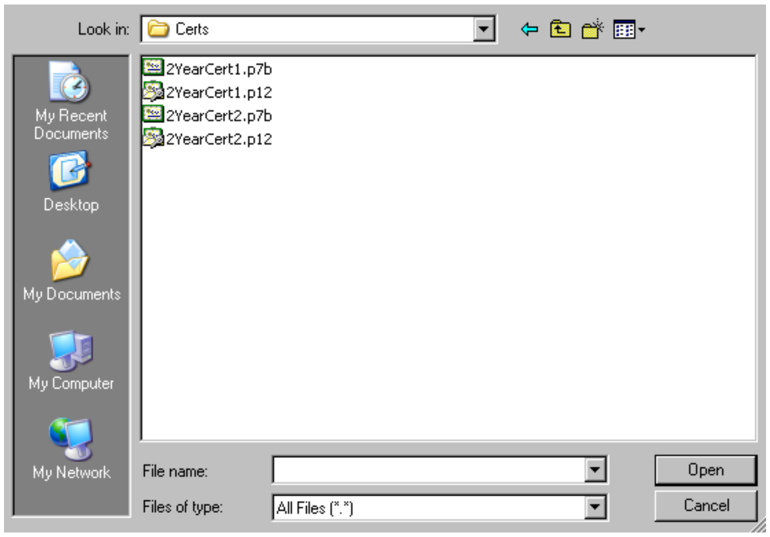
Enter the name of the personal certificate file to import. The file must have an extension of .pfx or .p12. The password is the one used to export the file from another application.

Certificate file:*

Password:

Figure 93. Certificate wizard locate the certificate file page

2 To locate the PKCS#12 file containing your certificate, click **Browse** to display the Browse dialog box.



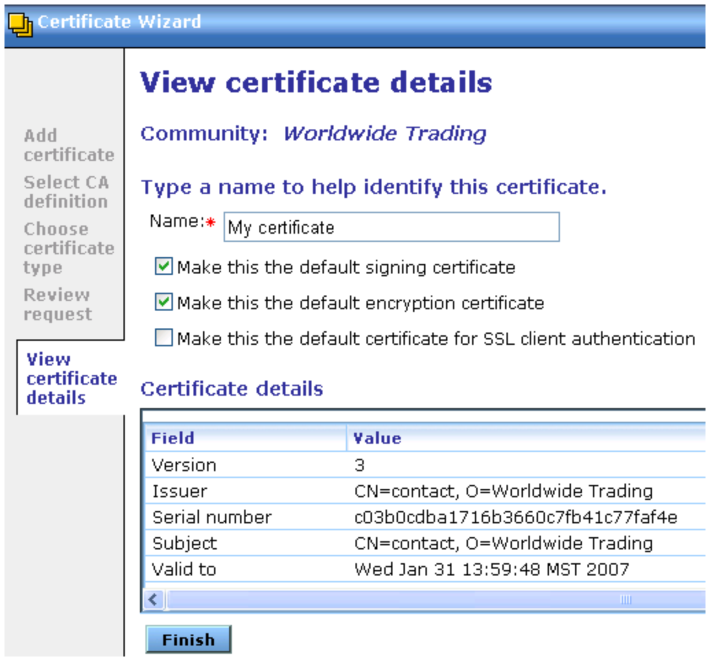
2YearCert1.p7b
2YearCert1.p12
2YearCert2.p7b
2YearCert2.p12

Figure 94. Browse dialog box

3 Locate and select the certificate file. The file must have an extension of .pfx or .p12. Click **Open** and the certificate file location certificate page reappears.

4 Type the same password you used when you exported the certificate file from a browser or mail client.

5 Click **Next** to display the certificate details page.



View certificate details

Community: *Worldwide Trading*

Type a name to help identify this certificate.

Name:* My certificate

☑ Make this the default signing certificate

☑ Make this the default encryption certificate

☐ Make this the default certificate for SSL client authentication

Certificate details

Field	Value
Version	3
Issuer	CN=contact, O=Worldwide Trading
Serial number	c03b0cdba1716b3660c7fb41c77faf4e
Subject	CN=contact, O=Worldwide Trading
Valid to	Wed Jan 31 13:59:48 MST 2007

Figure 95. Certificate wizard details page

6 If you want, type a name for the certificate in the Name field. This name can help you tell one certificate from another. By default the system uses the CA name as the certificate name.

To make the certificate your default signing certificate, click **Make this the default signing certificate**. This option is selected by default.

To make the certificate your default encryption certificate, click **Make this the default encryption certificate**. This option is selected by default.

To make the certificate your default SSL authentication certificate, click **Make this the default certificate for SSL client authentication**. This means the community will present this certificate to a partner who requests client authentication to connect to a SSL server. See SSL authentication on page 316.

7 Review the certificate information on the page. Click **Finish** to import the certificate.

After the certificate is imported, the certificates page reappears, displaying the new certificate.

8 If you are setting up a community profile for the first time, you must distribute your certificate information by sending it to partners on diskette or by some secure means. This can be done by exporting your certificate as part of your community profile. See Export a community profile on page 151.

If you need to distribute your certificate to your trading partners who use other interoperable software, see Export a certificate to a file on page 355.

Before you attempt to exchange encrypted and signed documents, you should contact each partner with whom you exchanged certificates and confirm that the fingerprints in both your certificates are identical. For more information see MD5 and SHA1 fingerprints on page 333.

Import certificates for partners

Use this procedure to import a partner's certificate and associate it with a partner profile.

If a partner also uses Cyclone Commerce software, the partner's certificate and public key normally is inside the imported partner profile. You need to import a partner's certificate file when:

- A partner uses other interoperable software and you must import the partner's certificate file after creating the partner's profile.

  or

- A partner sends you an updated certificate file to replace one that is associated with the partner's profile on your system.

If your partner uses other interoperable software and wants to send you a self-signed certificate, advise the partner to export the certificate to a PKCS#7 file (.p7c) and include all certificates in the certification path, if possible.

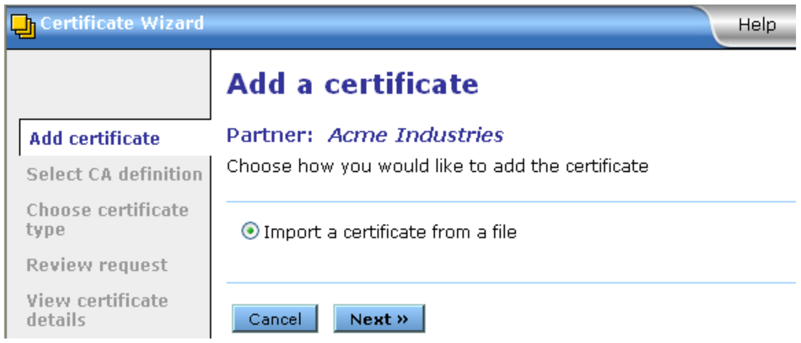
Steps

1 Make sure you have access on your system to the certificate file your partner sent you.

2 In the partners area of the user interface, go to the summary page for the partner you want.

3 Click **Certificates** on the navigation graphic at the top of the partner summary page to open the partner's certificates page.

4 Click **Add a certificate** to start the certificate wizard.



Add a certificate

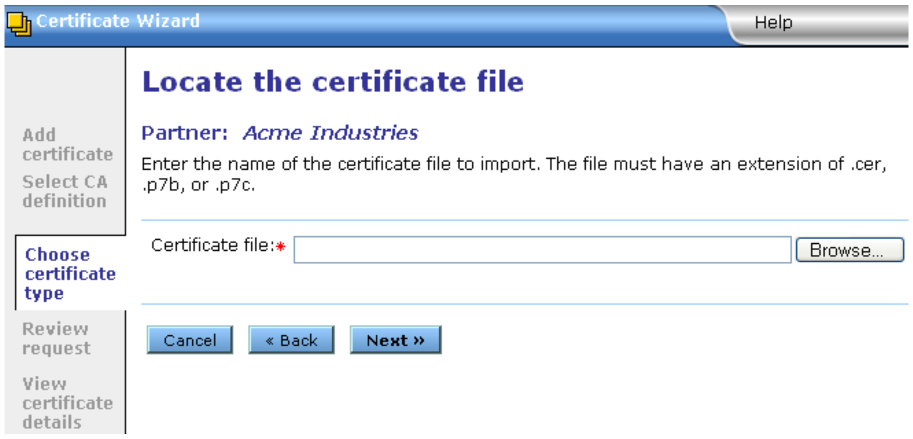
Partner: *Acme Industries*

Choose how you would like to add the certificate

● Import a certificate from a file

Figure 96. Certificate wizard import certificate from a file page

5 Click **Next** to display the file selection page.



Locate the certificate file

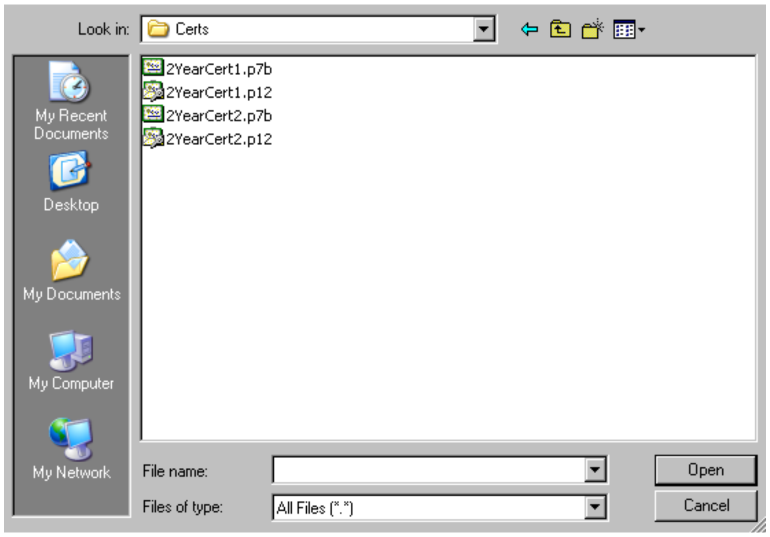
Partner: *Acme Industries*

Enter the name of the certificate file to import. The file must have an extension of .cer, .p7b, or .p7c.

Certificate file:*

Figure 97. Certificate wizard certificate file page

6 Click **Browse** to open the Browse dialog box.

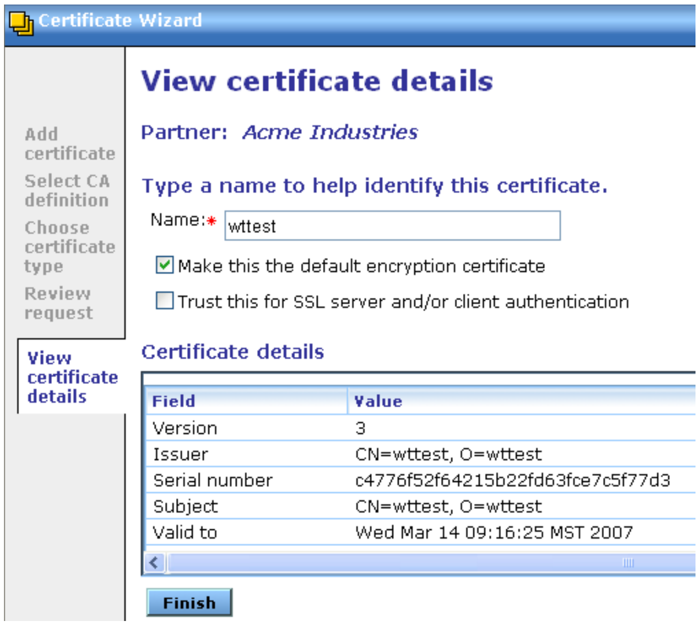


2YearCert1.p7b
2YearCert1.p12
2YearCert2.p7b
2YearCert2.p12

Figure 98. Browse dialog box

7 Select the certificate file you want to import and click **Open** to redisplay the certificate wizard.

8 Click **Next** to display the view certificate details page.



View certificate details

Partner: *Acme Industries*

Type a name to help identify this certificate.

Name:* wttest

[✓] Make this the default encryption certificate

[ ] Trust this for SSL server and/or client authentication

Certificate details

Field	Value
Version	3
Issuer	CN=wttest, O=wttest
Serial number	c4776f52f64215b22fd63fce7c5f77d3
Subject	CN=wttest, O=wttest
Valid to	Wed Mar 14 09:16:25 MST 2007

Figure 99. Certificate wizard view details page

9 If you want, type a name for the certificate in the Name field. This name can help you tell one certificate from another.

To make the certificate the default encryption certificate, select **Make this the default encryption certificate**. This option is selected by default.

To make the certificate the default SSL authentication certificate, select **Trust this for SSL server and/or client authentication**. This means the partner will present this certificate when the community requests client authentication to connect to a SSL server. See SSL authentication on page 316.

10 Click **Finish**. The partner certificates page is redisplayed with the certificate you imported.

11 Before you attempt to exchange encrypted and signed documents, contact the partner and confirm that the fingerprints in the certificate you imported are identical to the partner's. For more information see MD5 and SHA1 fingerprints on page 333.

Export a certificate to a file

Use this procedure to export a community or partner certificate to a file. The procedure for both is similar, except for a community certificate you have the option of exporting the private key as well as the public key. This option does not apply to partner certificates, which only contain public keys.

After exporting a community certificate, you can send the file to your partners by e-mail or on diskette.

For your partner, export a community certificate that contains your public key. Never give your partner a community certificate that contains your private key.

For yourself for backup purposes, you can export a community certificate that contains your private and public keys. If you do, keep this certificate in a secure place and never give it to anyone.

You might want to export a partner certificate and public key to a file to keep as a backup. If the partner certificate is deleted from the system, you can import the certificate again if needed.

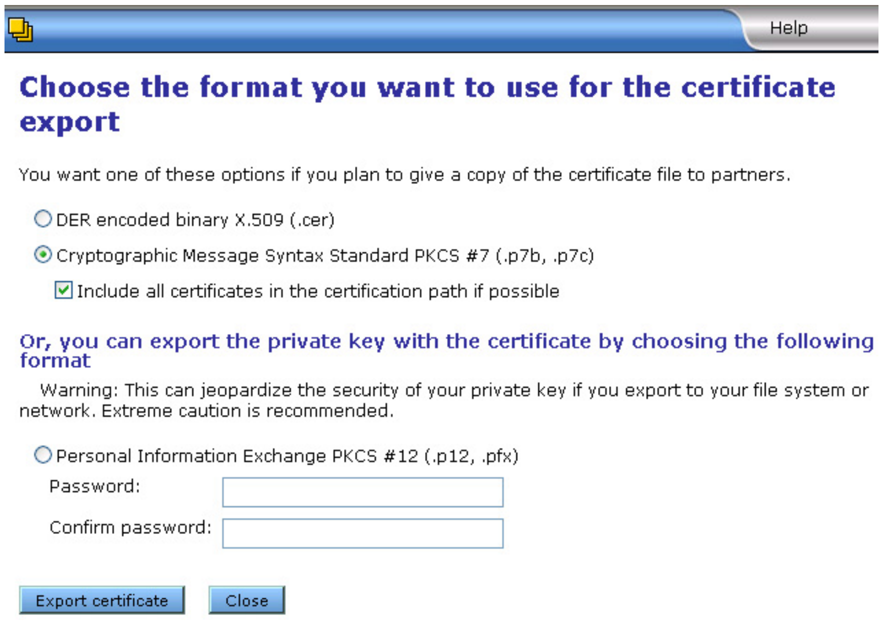
Steps

1 In the community or partner area of the user interface, go to the summary page for the community or partner you want.

2 Click **Certificates** on the navigation graphic at the top of the community or partner summary page to open the certificates page.

3 Click the name of the certificate to export to open the certificate information page.

4 Click **Export this certificate** to open the certificate export page. Figure 100 shows a community certificate export page. A partner certificate export page does not have the option for exporting a private key.



Choose the format you want to use for the certificate export

You want one of these options if you plan to give a copy of the certificate file to partners.

○ DER encoded binary X.509 (.cer)

● Cryptographic Message Syntax Standard PKCS #7 (.p7b, .p7c)

[✓] Include all certificates in the certification path if possible

Or, you can export the private key with the certificate by choosing the following format

Warning: This can jeopardize the security of your private key if you export to your file system or network. Extreme caution is recommended.

○ Personal Information Exchange PKCS #12 (.p12, .pfx)

Password:

Confirm password:

Figure 100. Export certificate page

5 Select an export option. If you are exporting a community certificate for use by a partner, note that the DER and PKCS#7 options are functionally the same. However, the one to select depends primarily on what your partner's trading engine supports.

For trading between partners who both use Cyclone Commerce software, we recommend selecting PKCS#7 and the check box for include all certificates in the certification path. Although this is the most all-inclusive choice, you can nevertheless choose DER instead with no adverse effects.

The following explains the options in more detail. If you trade with partners who use interoperable software, we recommend that you determine whether their software supports DER, PKCS#7 or both.

DER encoded binary X.509 (.cer)

Select this option to export a binary file with an extension of cer. The file contains a single binary certificate containing a public key.

If your partner's software only supports DER encoded certificates, select this option.

Cryptographic Message Syntax Standard PKCS #7 (.p7b, .p7c)

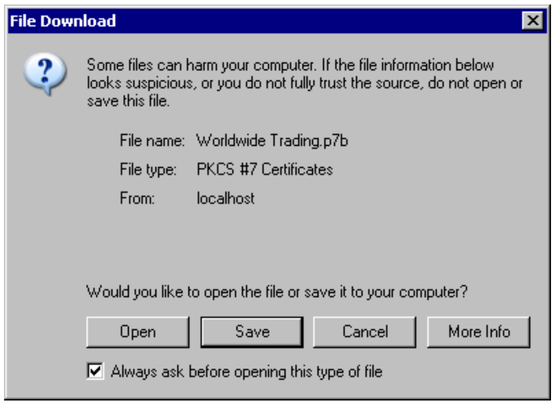
Select this option to export a file with an extension of p7c. The file can contain all the certificates needed to support trading, if more than one is required. If your partner's software supports a certificate in a PKCS#7 format, we recommend this option over DER.

If you select this, you also can select **Include all certificates in the certification path if possible**. This option includes all certificates in the chain of trust for the certificate. This is the most all-inclusive method for exporting a certificate. However, be aware that your partner's software, if not Cyclone Commerce, might not support the entire certificate path in the p7c file.

Personal Information Exchange PKCS #12 (.p12, .pfx)

Select this option to export a community certificate containing the private key. You should do this only if you can keep the certificate in a highly secure place. This option is available only for community certificates and not partner certificates. Your user role must have permissions to export private keys.

6 Click **Export certificate** to display a file download dialog box.


File Download

Some files can harm your computer. If the file information below looks suspicious, or you do not fully trust the source, do not open or save this file.

File name: Worldwide Trading.p7b

File type: PKCS #7 Certificates

From: localhost

Would you like to open the file or save it to your computer?

☑ Always ask before opening this type of file

Figure 101. File download dialog box

7 Click **Save** to display the Save As dialog box.

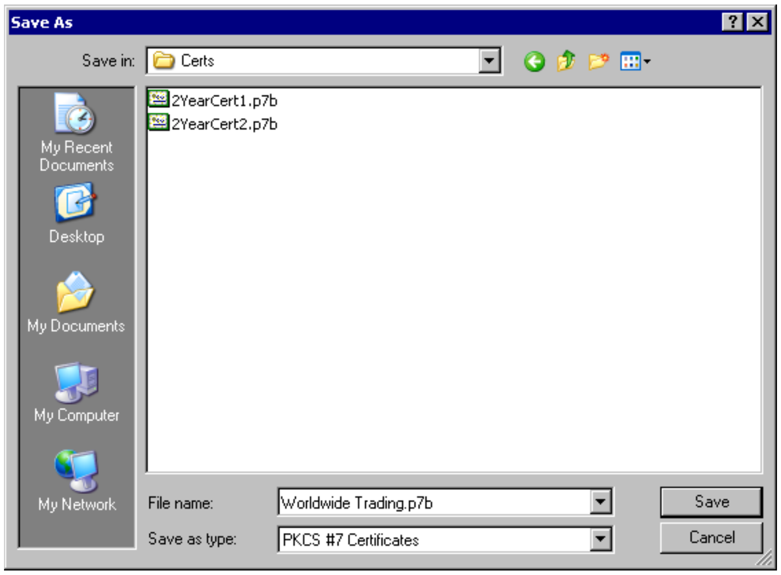


Figure 102. Save As dialog box

8 Review the file name and path for the file you are exporting. If you want to change the path or name, you can navigate to a new folder or type a new file name.

9 Click **Save** to export the certificate. The certificate export page reappears.

10 When a download complete message displays, click **Close**.

11 If you exported a community certificate for a partner, send the certificate file to the partner by a secure means.

Delete certificate

Use this procedure to delete certificates that you or your partners no longer use for verifying signatures, encrypting messages or SSL authentication.

Once you delete a certificate, you cannot retrieve it. If you find that you need a deleted certificate, you must import it from a file.

Steps

1 On the certificates page, select the certificate you want to delete and click **Delete this certificate**. A dialog box appears with a message asking whether you want to delete the certificate.

2 Click **Yes** to delete the certificate or **No** to cancel the operation.

Certificate revocation lists

You can have the trading engine compare certificates against lists of invalid certificates that are maintained by the issuing certificate authorities. This checking is done for outbound encrypted documents and inbound signed documents.

A certificate revocation list (CRL) is a list of third-party certificates that are no longer valid. Certificate authorities maintain such lists of certificates they issued but later invalidated for one reason or another. CRLs are accessible on the Internet, and you need an Internet connection to use them.

Actually, the trading engine always attempts to do CRL checking for each traded message. Whether it succeeds in the attempt, and whether it fails a message with a revoked certificate, depends on whether a CRL is available to check a certificate against. CRL checking is only attempted for CA-issued certificates, never for self-signed certificates.

A certificate is checked if a CRL can be obtained from a CA. If there is a CRL for a given certificate, that CRL is issued and signed by the same issuer as the certificate. The CRL may have been downloaded from a distribution point that matches one specified in a CRL distribution point extension in the certificate. Most CRLs have a **thisUpdate** time indicating when the CRL was last updated by its issuer. Most also have a **nextUpdate** time indicating when the issuer will next update the CRL. A CRL may be updated before its nextUpdate time.

As an example of CRL checking, when a partner sends you an encrypted document, the trading engine checks the certificate associated with the inbound document against the appropriate CRL. If the certificate is on the CRL, the trading engine fails the inbound document.

The trading engine checks a specific certificate only against the appropriate CRL. For example, the trading engine will not check a certificate issued by VeriSign against a CRL issued by GlobalSign.

Although using CRLs can enhance security, the checking process can result in longer processing times. Consequently, your decision whether to use CRLs should weigh the security advantage against the performance handicap.

The trading engine usually checks a certificate against a previously downloaded CRL. But this can be extended to on-the-fly CRL downloading and checking by setting the **crls.autoRetrieve** property in the crossworks.properties file to **true**. With this enabled, the trading engine looks for a CRL distribution point URL in the certificate being checked and downloads the updated CRL if available. It then checks the certificate against the newly downloaded CRL. If an updated CRL is not available, the trading engine checks the certificate against the previously downloaded CRL. Because of this on-the-fly downloading and checking capability, there can be multiple CRLs from the same issuer. For more about how to tune CRL checking, see Advanced CRL settings on page 364.

You are responsible for obtaining from the certificate authority the information required for accessing the CRL. The trading engine downloads the latest CRL before performing certificate checks. It also can download updates of the CRL, based on the update interval in the previously downloaded CRL.

How CRL checking works

CRL checking is potentially performed for every non-root certificate encountered during certificate path validation. The following steps are taken for each such certificate.

1. The system retrieves the CRL distribution point, if any, from the current certificate. The CRL distribution point is the first URL found in a CRL distribution points extension in the certificate. Not all certificates have a CRL distribution point extension and not all such extensions have a URL.

2. The system looks in its cache of CRLs for an effective CRL with the same issuer and CRL distribution point (if any) as the current certificate. A CRL is considered to be effective if the current time is between the CRL's thisUpdate and nextUpdate times. If such a CRL is found, it is used: Go to step 9.

3. If the **crls.autoRetrieve** property is set to **true** in crossworks.properties and the certificate contains a CRL distribution point, the system attempts to retrieve a CRL from that distribution point. If a CRL is successfully retrieved, it is added to the CRL cache. The system again looks in its cache of CRLs for an effective CRL with the same issuer and CRL distribution point as the current certificate. If such a CRL is found, it is used. Go to step 9.

4. If the **crls.useExpired** property is set to **true** in crossworks.properties, the system looks in its cache of CRLs for the most recently expired CRL with the same issuer and CRL distribution point (if any) as the current certificate. A CRL is considered to be expired if its nextUpdate time is in the past. If such a CRL is found, it is used. Go to step 9.

5. If the current certificate does not contain a CRL distribution point, there is no CRL to check. Go to step 8.

6. The system looks in its cache of CRLs for an effective CRL with the same issuer as the current certificate. If such a CRL is found, it is used. Go to step 9.

7. If the crls.useExpired property is set to true, the system looks in its cache of CRLs for the most recently expired CRL with the same issuer as the current certificate. If such a CRL is found, it is used. Go to step 9. Note that this step is similar to step 4, with the exception that here the CRL distribution point is not checked.

8. No CRL could be found for checking the current certificate. If the crls.require property is set to true, the certificate path validation fails. Otherwise, the CRL check for the current certificate succeeds.

9. The signature of the found CRL is verified using the CRL's issuing certificate. If the verification fails, the certificate path validation fails.

10. The list of certificates in the CRL is checked whether it contains the current certificate. If the certificate is listed in the CRL, the certificate path validation fails. Otherwise, the CRL check for the current certificate succeeds.

Obtaining CRL access information

Before you can download a CRL to the trading engine, you must obtain the information required to access the CA’s CRL.

CRLs are usually made available via one of two protocols: hypertext transfer protocol (HTTP) and lightweight directory access protocol (LDAP). For example, VeriSign CRLs are accessed via HTTP and Entrust CRLs are accessed via LDAP.

You can obtain the CRL information by viewing the details of a CA-issued certificate. See Details tab on page 332. The information, if present, is labeled as a CRL distribution point.

As an example, the following is the CRL distribution point within a VeriSign certificate. This is a URL as follows:

http://crl.verisign.com/class1.crl

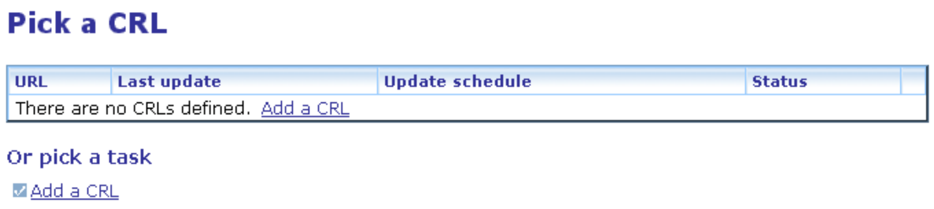
You would enter this URL to add a VeriSign CRL to the trading engine.

Add a CRL

Use this procedure to download a CRL. For details about CRLs, see Certificate revocation lists on page 359.

Steps

1 On the system management page, click **Manage CRLs** to display the CRL list page. The page lists imported CRLs, if any.


Pick a CRL

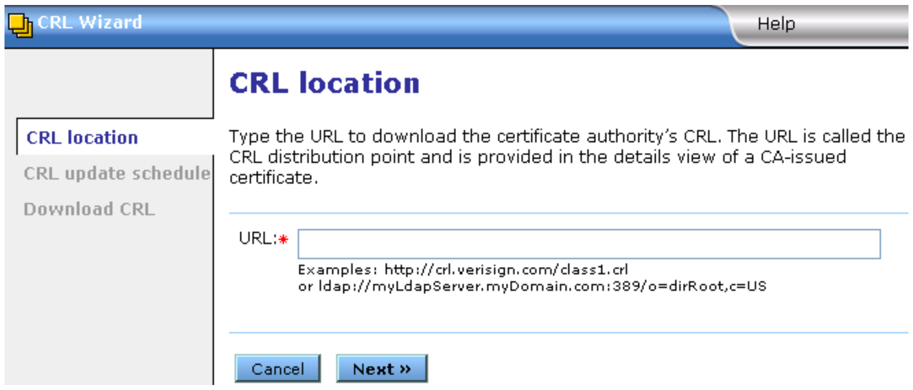
URL	Last update	Update schedule	Status
There are no CRLs defined. Add a CRL			

Or pick a task

Add a CRL

Figure 103. Certificate revocation list page

2 Click **Add a CRL** to launch the CRL wizard.



CRL location

Type the URL to download the certificate authority's CRL. The URL is called the CRL distribution point and is provided in the details view of a CA-issued certificate.

URL:*

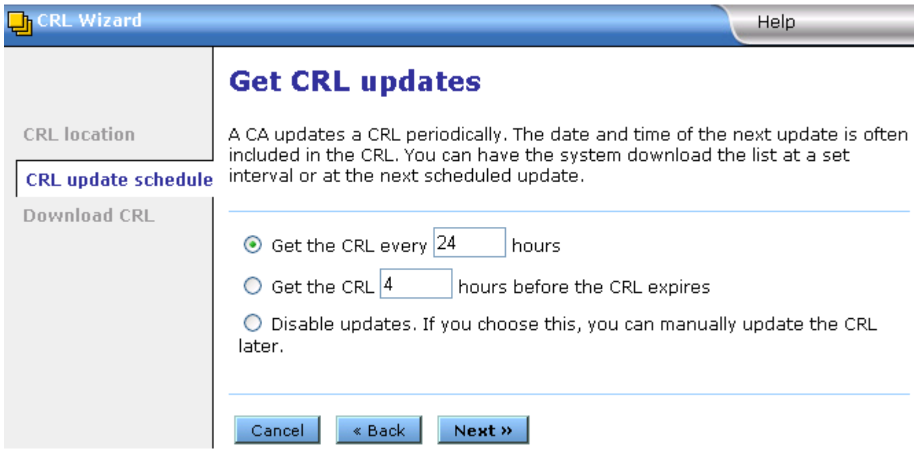
Examples: http://crl.verisign.com/class1.crl
or ldap://myLdapServer.myDomain.com:389/o=dirRoot,c=US

Figure 104. CRL wizard CRL location page

3 Type the URL to download the CRL. See Obtaining CRL access information on page 361.

If you must connect through a proxy server to retrieve a CRL, see the proxy server properties described in Advanced CRL settings on page 364.

4 Click **Next** to display the get CRL updates page.



Get CRL updates

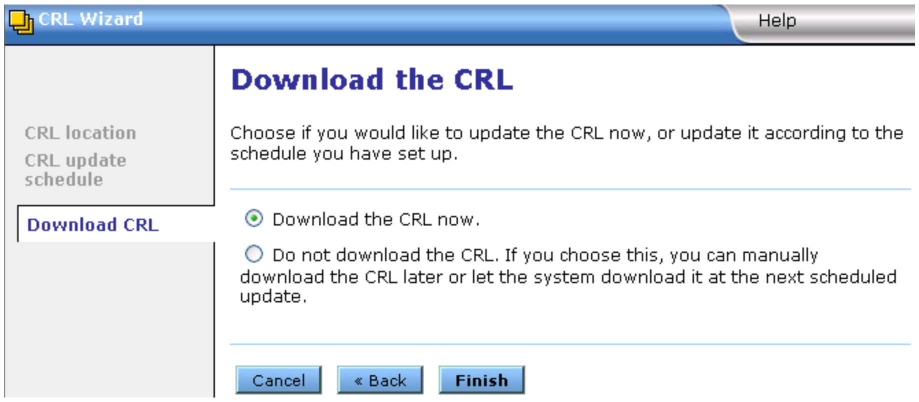
A CA updates a CRL periodically. The date and time of the next update is often included in the CRL. You can have the system download the list at a set interval or at the next scheduled update.

[✓ Get the CRL every 24 hours]
Get the CRL 4 hours before the CRL expires
Disable updates. If you choose this, you can manually update the CRL later.

Figure 105. CRL wizard get CRL updates page

5 Select the interval for downloading an updated CRL.

6 Click **Next** to display the download the CRL page.



Download the CRL

Choose if you would like to update the CRL now, or update it according to the schedule you have set up.

[✓ Download the CRL now.]
Do not download the CRL. If you choose this, you can manually download the CRL later or let the system download it at the next scheduled update.

Figure 106. CRL wizard download the CRL page

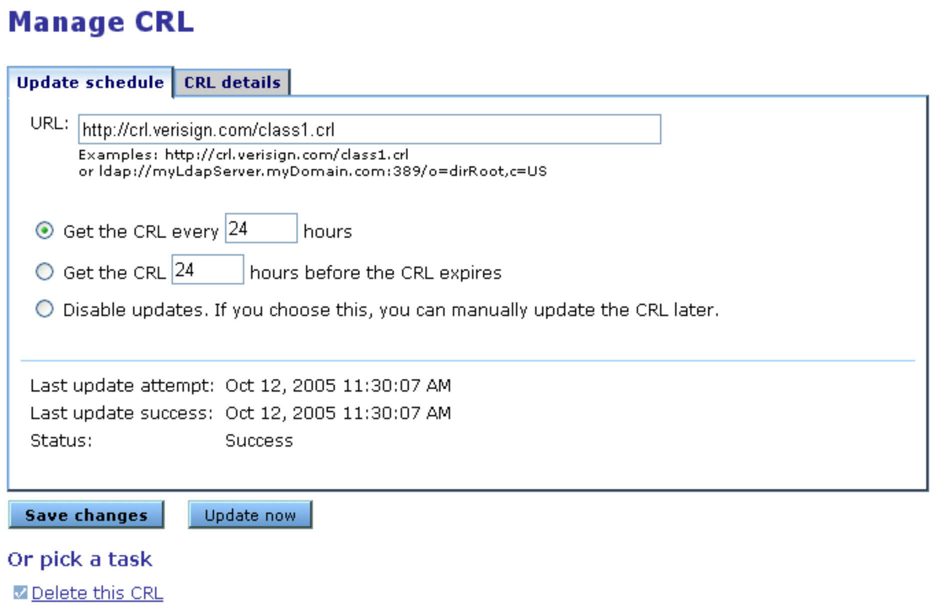
7 Select whether you want to download the CRL now or later.

8 Click **Finish**. If you elected to download the CRL now, the CRL is listed on the CRL list page with a status of **Success**. CRLs are stored in [install directory]\common\conf\crls.

Manage CRLs

Once one or more CRLs have been downloaded, you can use the CRL list page to manage them. To open this page, go to the system management area of the user interface and click **Manage CRLs**. The CRL list page shows the name of each CRL, the date and time the CRL was last updated, the update schedule status, and the download status.

On this page you can click the name of a CRL and open another page that lets you change the URL for downloading the CRL or the updating interval. You also can view details of the CRL.


Manage CRL

URL: http://crl.verisign.com/class1.crl

Examples: http://crl.verisign.com/class1.crl
or ldap://myLdapServer.myDomain.com:389/o=dirRoot,c=US

[✓ Get the CRL every 24 hours]
Get the CRL 24 hours before the CRL expires
Disable updates. If you choose this, you can manually update the CRL later.

Last update attempt: Oct 12, 2005 11:30:07 AM
Last update success: Oct 12, 2005 11:30:07 AM
Status: Success

Figure 107. Manage CRL page

To delete a CRL, click **Delete** to the right of the CRL name on the CRL list page or click **Delete this CRL** on the manage CRL page. Deleted CRLs no longer appear on the CRL list page and are deleted from [install directory]\common\conf\crls.

Advanced CRL settings

The retrieval and usage of CRLs are controlled by properties in the crossworks.properties file. This file is in [install directory]\[build number]\conf.

For most users, the file's default settings for the crls.useExpired, crls.require and crls.autoRetrieve properties are sufficient. But users with special needs may want to change some settings. Note that when the default values are used and no current CRLs are in [install directory]\common\conf\crls, CRL checking does not occur.

The following describes the CRL properties in crossworks.properties. The trading engine periodically checks whether crossworks.properties has changed and re-loads its contents if necessary. There is no need to restart the server upon changing the file, but allow up to a minute for changes made to the file to take effect.

crls.useExpired

Set to **true** to allow using an expired CRL when the trading engine cannot find a current CRL in [install directory]\common\conf\crls. The trading engine looks for a current CRL with the same issuer name as the certificate being checked. In a current CRL, the current time stamp is between the CRL's thisUpdate and nextUpdate time stamps. If the trading engine cannot find such a CRL, it looks for a CRL with the same issuer name as the certificate and with a nextUpdate time stamp that is before the current time stamp. If more than one such expired CRL is found, the most recently expired CRL is used to check whether the certificate has been revoked.

Set to **false** to use only CRLs that have not expired.

The default is **false**.

crls.require

Set to **true** to require CRL checking for all non-root (not self-signed) certificates and to fail certificate path validation when the trading engine cannot find a certificate's CRL. Normally, if a CRL cannot be found for a certificate, the certificate is presumed valid and certificate path validation continues. When crls.require is set to **true**, the trading engine must find a CRL for each non-root certificate. If a CRL is not found, the certificate is considered revoked and the certificate path validation fails.

When set to **true**, make sure update schedules are set on the manage CRL page for all CRLs needed to check non-root and intermediary certificates in the certificate path.

Set to **false** to continue validation when a CRL is not found for a non-root certificate.

The default is **false**.

crls.autoRetrieve

Indicates whether CRLs should be automatically retrieved using the information in certificates' CRL distribution points extension.

When **crls.autoRetrieve=true** and the trading engine does not already have an effective CRL for a certificate, it will attempt to use the certificate's CRL distribution point extension to retrieve the CRL for that certificate. Note that failure to retrieve a CRL will result in failure to validate the certificate if the **crls.require** property also is set to **true**.

The default is **false**.

crls.http.proxyHost

The name or IP address of the proxy server to use when retrieving CRLs via HTTP.

crls.http.proxyPort

The port number of the proxy server to use when retrieving CRLs via HTTP. Only used if crls.http.proxyHost is set. If not set, defaults to 80.

crls.http.proxyUser

The user name to use for authentication with the proxy server when retrieving CRLs via HTTP. Only used it crls.http.proxyHost is set.

crls.http.proxyPassword

The password to use for authentication with the proxy server when retrieving CRLs via HTTP. Only used if crls.http.proxyUser is set. If not set, defaults to no password.

❖ ❖ ❖

19 Collaboration settings

The trading engine lets you configure how it packages the messages a community sends. These collaboration settings control whether messages are sent in cipher or plain text, and whether partners must acknowledge receiving messages.

These settings only affect how messages are sent, not received. Rules for receiving messages are set elsewhere (see Inbound message validation on page 399).

Concepts

- Hierarchy of outbound settings on page 367
- Community collaboration settings on page 391
- Partner collaboration settings on page 394

Procedure

- View settings between partners on page 368
- View or change default settings on page 370

Pages and fields

- Default collaboration settings on page 371

Hierarchy of outbound settings

There are three possible levels of collaboration settings that are applied in hierarchical order. You can use only one level — the default settings for all communities — or all three. The user interface depicts the hierarchical relationship. Figure 108 illustrates them. The levels are:

1. **Default collaboration settings for all communities**

   These settings apply to all communities and all partners unless superseded. You can change these global settings as you wish.

2. **Collaboration settings for one community**

   A community can establish its own collaboration settings. These override any global defaults that are different. These settings apply to all of a community's partners unless superseded.

3. Collaboration settings for one partner

A community can establish collaboration settings that apply to only one partner. These settings override any other settings that are different.

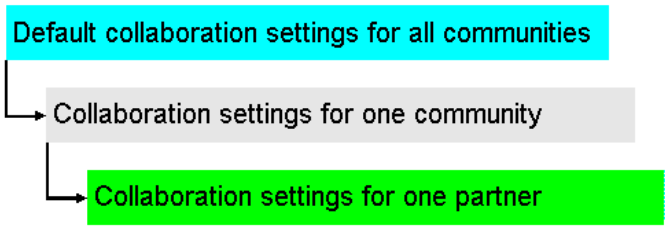

Default collaboration settings for all communities
Collaboration settings for one community
Collaboration settings for one partner

Figure 108. Hierarchy of collaboration settings

View settings between partners

Use this procedure to view the collaboration settings between one community and one partner.

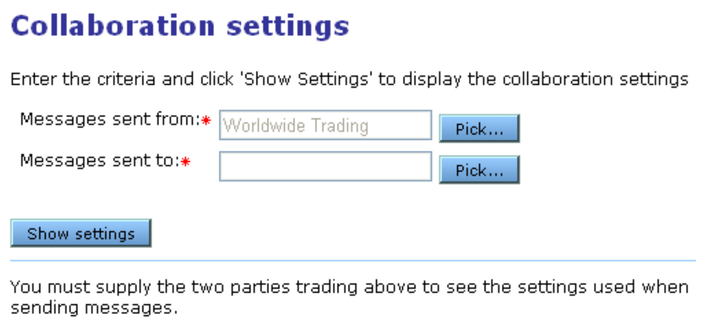
Viewing collaboration settings between parties is a way to check whether delivery exchanges are properly set up. Although this does not rule out possible infrastructure or configuration issues that could affect trading, it does help ensure that two parties' security and packaging preferences are correct for outbound payloads and receipts from partners.

Before checking collaboration settings between two parties, make sure your community profile and the partner profile use the same message protocol. That is, the community and partner profiles both must have the same type of message protocol set up.

In addition, the partner protocol transport must be the profile's default exchange. This means the transport must be the first listed. The trading engine uses the default protocol to send messages to partners. Click **Delivery exchange** on the navigation graphic on the partner profile summary page to check. The only exception is a transport for the ebXML message protocol. It must be present, but does not have to be the default transport.

Steps

1 On the main trading configuration page or a community summary page, click **Show collaboration settings** to display the collaboration settings page.



Collaboration settings

Enter the criteria and click 'Show Settings' to display the collaboration settings

Messages sent from:* Worldwide Trading

Messages sent to:*

You must supply the two parties trading above to see the settings used when sending messages.

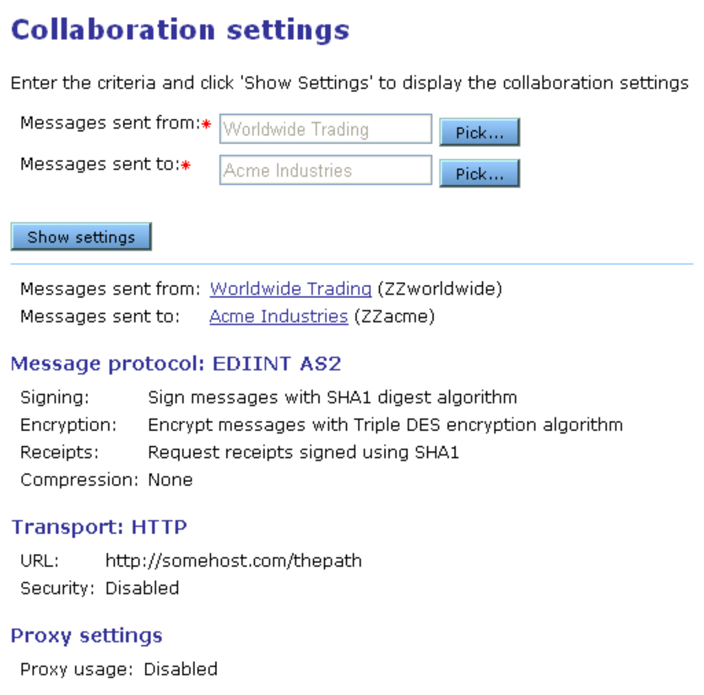
Figure 109. Collaboration settings page

If you navigated from a community summary page, the page by default uses the current community as the sender. You can use the default or change it.

2. Using the **Pick** buttons, select the community sender (sent from) and the partner receiver (sent to).

3. Click **Show settings**. The system searches for the collaboration settings. It checks for any settings specific to the community or partner that override the default global settings. The page refreshes with the settings for the community and partner.

   If the system reports a collaboration could not be created between the parties, a common failing is the message protocols in the community and partner profiles are not the same (for example, not both AS2).



Collaboration settings

Enter the criteria and click 'Show Settings' to display the collaboration settings

Messages sent from:*	Worldwide Trading
Messages sent to:*	Acme Industries

Messages sent from: Worldwide Trading (ZZworldwide)
Messages sent to: Acme Industries (ZZacme)

Message protocol: EDIINT AS2

Signing:	Sign messages with SHA1 digest algorithm
Encryption:	Encrypt messages with Triple DES encryption algorithm
Receipts:	Request receipts signed using SHA1
Compression:	None

Transport: HTTP

URL:	http://somehost.com/thepath
Security:	Disabled

Proxy settings

Proxy usage: Disabled

Figure 110. Collaboration settings page with positive results

View or change default settings

Use this procedure to view or change the default settings for outbound messages. These apply to all communities, unless superseded by community- or partner-level settings. See Hierarchy of outbound settings on page 367.

Steps

1 On a community summary page, click **Collaboration settings** on the navigation graphic at the top of the page. This opens the community's collaboration settings page.

2 Click **Default settings** at the top left of the page. This opens the defaults for sending messages page. The page is organized by tabs for outbound message protocols. You only need to pay attention to the tabs for the protocols you use. There also is a reliable messaging tab, which controls message resends.

3 To view or change settings, click the tab you want. If you change anything, click **Save changes**.

See Default collaboration settings on page 371 for descriptions of the fields on each tab.

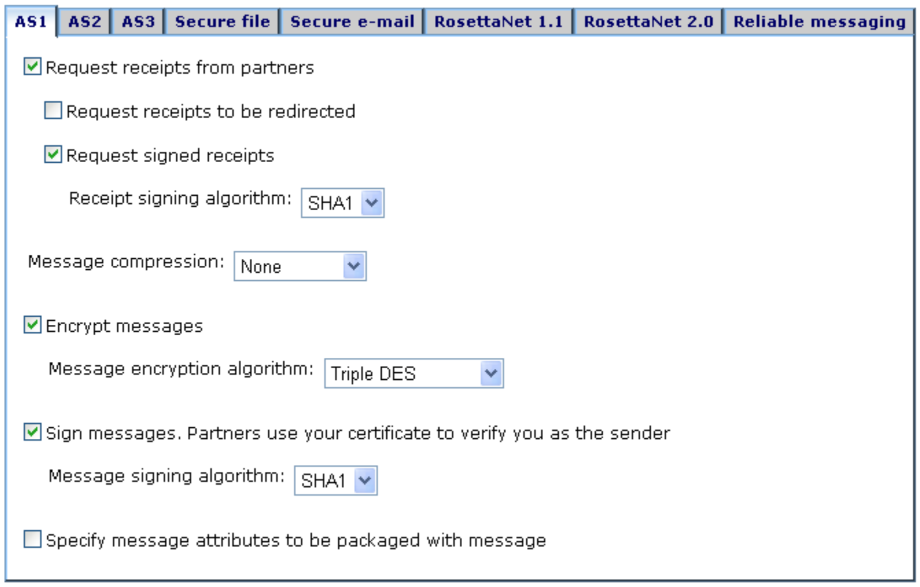
Default collaboration settings

The following topics describe the default collaboration settings. These control how messages are sent for each message protocol. There also are settings for reliable messaging, which controls message resends.

For procedure see View or change default settings on page 370.

AS1 default settings

The AS1 tab shows default outbound message settings for the AS1 message protocol. The settings affect all outbound documents for all communities, unless superseded by community- or partner-level settings. You can change these as you need. Figure 111 displays the default AS1 settings tab.



☑ Request receipts from partners

☐ Request receipts to be redirected

☑ Request signed receipts

Receipt signing algorithm: SHA1

Message compression: None

☑ Encrypt messages

Message encryption algorithm: Triple DES

☑ Sign messages. Partners use your certificate to verify you as the sender

Message signing algorithm: SHA1

☐ Specify message attributes to be packaged with message

Figure 111. Default AS1 settings

Request receipts from partners

Select this check box to have your partners send receipts to you upon receiving your messages. The receipts are signed or unsigned, depending on your selection in the request signed receipts check box.

Request receipts to be redirected

Select this check box to request that receipts sent by partners use the address in the Receipt-Delivery-Option line in the MIME header of outbound AS1 messages. You must type the alternate address in the return e-mail address field.

Request signed receipts

Select this check box to have your partners sign the receipts they send you.

Receipt signing algorithm

This is the algorithm you want partners' trading engines to use when creating a hash of the unencrypted receipt. This hash is a number that is encrypted with the partner's private key. It is decrypted by the community using the partner's public key. The community rehashes the decrypted receipt and compares the result with the hash that came with the receipt. If the two are identical, it ensures the contents have not been altered.

Message compression

Select a method for compressing the messages you send. This is optional.

Compressing data before sending increases throughput. You might want to consult with your partner about compressing and uncompressing as pre- and post-processing steps. You and your partner can compress or not independently, as long as the receiver's system supports uncompressing.

The compression options are:

None. Select this if you do not want compression. Also, you must select this to exchange messages with partners whose interoperable software cannot uncompress documents.

EDIINT/ZLIB is for trading with partners who use version 4.2 or later of Cyclone Activator or who use an EDIINT-compliant trading engine other than Cyclone Activator.

MIME/GZIP is for trading with partners who use versions of Cyclone Activator earlier than 4.2. You also can select this if your partner uses a trading engine other than Cyclone Activator that supports GZIP compression.

Encrypt messages

Select this check box to encrypt the messages you send.

Message encryption algorithm

If you select encrypt messages, select an algorithm.

Sign messages. Partners use your certificate to verify you as the sender

Select this check box to digitally sign the messages you send.

Message signing algorithm

The algorithm for creating a hash of the unencrypted message. This hash is a number that is encrypted with your community's private key. It is decrypted by the partner using the community's public key. The partner rehashes the decrypted message and compares the result with the hash that came with the message. If the two are identical, it ensures the contents have not been altered.

Specify message attributes to be packaged with message

This option is available only if your license has a **true** permission for allowCustomMetadataInMessageHeaders.

Checking this enables you to include any meta-data elements you want in the MIME headers of the messages your community sends. This can serve several purposes. It lets your community forward payload-related information to a partner. It also lets a community or partner use such data to trigger message handling actions.

The meta-data that can be added are in addition to the meta-data already in MIME headers, such as sender and receiver, content type, disposition notification and so on.

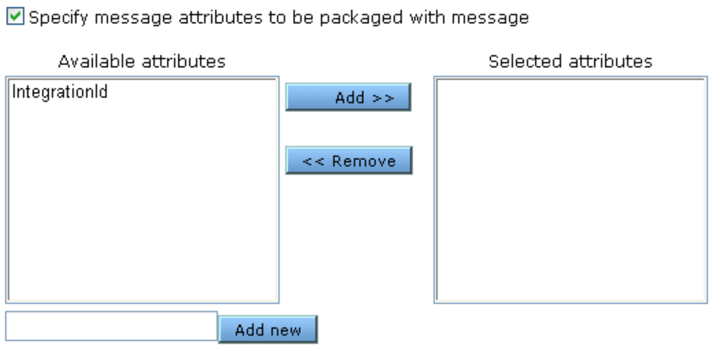


Figure 112. Available and selected attributes list boxes

There are several ways to add meta-data elements to the available attributes list on this tab. You can type an attribute name in the field below the available attributes list and click **Add new**. Or, you can go to the message handler area of the user interface and add an attribute (see the chapter on message handling). Any attributes added in the message handler area appear on the available attributes list on this tab.

We recommend these guidelines for attribute names: Make attribute names a single text string. For names that use multiple words, do not use spaces between the words. Use camel case for names that include two or more words (for example, AttributeName). Do not use non-alphanumeric characters in attribute names (for example, commas, periods, hyphens, asterisks, ampersands and so on).

To add attributes to headers of outbound documents, use the **Add** button on this tab to move a selected attribute from the available attributes list to the selected attributes list. Click **Save changes** when done.

In addition to selecting meta-data elements to include in headers, you must separately establish the values for the attributes. There are several ways to do this, including:

Message attributes tab. Use the message attributes tab on a transport maintenance page to set attribute values by use of directory mapping or fixed values. For more information see the message attributes tab topic in the transport maintenance chapter.

Message handler. In the message handler area, you can set a fixed value for an attribute. If the payload is an XML document, the value can be parsed by specifying an XPath. For more information see the chapter on message handling.

Inline processing. If you are a licensed SDK user, you can build a Java class and use it to provide attribute values or perform other message actions.

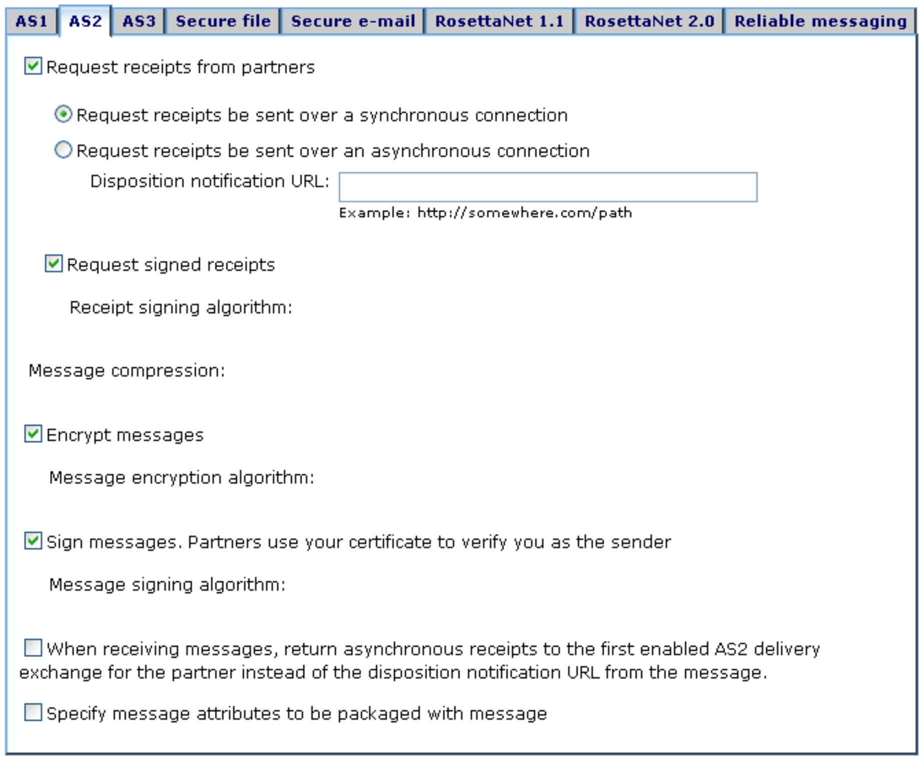
Upon receiving and unpackaging a message with extra meta-data, a partner who uses Cyclone Activator 5.3.2 or later can use the added-meta-data for post processing, inline processing or file system integration. Inform the partner of the meta-data elements you add to outbound messages.

The added meta-data in the header of the outbound document is in the following format:

X-Cyclone-Metadata-AttributeName: Value

AS2 default settings

The AS2 tab shows default outbound message settings for the AS2 message protocol. The settings affect all outbound documents for all communities, unless superseded by community- or partner-level settings. You can change these as you need. Figure 113 displays the default AS2 settings tab.



☑ Request receipts from partners

● Request receipts be sent over a synchronous connection

○ Request receipts be sent over an asynchronous connection

Disposition notification URL:

Example: http://somewhere.com/path

☑ Request signed receipts

Receipt signing algorithm:

Message compression:

☑ Encrypt messages

Message encryption algorithm:

☑ Sign messages. Partners use your certificate to verify you as the sender

Message signing algorithm:

☐ When receiving messages, return asynchronous receipts to the first enabled AS2 delivery exchange for the partner instead of the disposition notification URL from the message.

☐ Specify message attributes to be packaged with message

Figure 113. Default AS2 settings

Request receipts from partners

Select this check box to have your partners send receipts to you upon receiving your messages. The receipts are signed or unsigned, depending on your selection in the request signed receipts check box.

Request receipts be sent over a synchronous connection

Select this check box if you want synchronous receipts in accord with the AS2 standard.

A synchronous receipt is returned to the sender during the same HTTP session as the sender's original message.

If you trade messages larger than 10 megabytes, we recommend sending receipts over an asynchronous connection to avoid tying up system resources.

If you have a partner who uses a version of the trading engine earlier than 5.0, the default setting is asynchronous. Coordinate with your partner to make sure both of you have the same setting.

Request receipts be sent over an asynchronous connection

Select this check box if you want asynchronous receipts. If you select this, you must complete the next field.

An asynchronous receipt is returned to the sender on a different communication session than the sender's original message session.

If you trade messages larger than 10 megabytes, we recommend sending receipts over an asynchronous connection to avoid tying up system resources.

Disposition notification URL

Type the URL for the partner to use when sending asynchronous receipts to acknowledge receiving payloads from your community. You must type a URL if you request receipts over an asynchronous connection. The system does not provide this value for you. This can be the URL for a community inbound HTTP or HTTPs transport. It also can be the address of a community inbound e-mail transport. Refer to the community inbound transports to obtain the URL information.

If you choose to use an e-mail address, it must be in URL format. For example, mailto:community@mail.com.

Request signed receipts

Select this check box to have your partners sign the receipts they send you.

Receipt signing algorithm

This is the algorithm you want partners’ trading engines to use when creating a hash of the unencrypted receipt. This hash is a number that is encrypted with the partner’s private key. It is decrypted by the community using the partner’s public key. The community rehashes the decrypted receipt and compares the result with the hash that came with the receipt. If the two are identical, it ensures the contents have not been altered.

Message compression

Select a method for compressing the messages you send. This is optional.

Compressing data before sending increases throughput. You might want to consult with your partner about compressing and uncompressing as pre- and post-processing steps. You and your partner can compress or not independently, as long as the receiver’s system supports uncompressing.

The compression options are:

None. Select this if you do not want compression. Also, you must select this to exchange messages with partners whose interoperable software cannot uncompress documents.

EDIINT/ZLIB is for trading with partners who use version 4.2 or later of Cyclone Activator or who use an EDIINT-compliant trading engine other than Cyclone Activator.

MIME/GZIP is for trading with partners who use versions of Cyclone Activator earlier than 4.2. You also can select this if your partner uses a trading engine other than Cyclone Activator that supports GZIP compression.

Encrypt messages

Select this check box to encrypt the messages you send.

Message encryption algorithm

If you select encrypt messages, select an algorithm.

Sign messages. Partners use your certificate to verify you as the sender

Select this check box to digitally sign the messages you send.

Message signing algorithm

The algorithm for creating a hash of the unencrypted message. This hash is a number that is encrypted with your community’s private key. It is decrypted by the partner using the community’s public key. The partner rehashes the decrypted message and compares the result with the hash that came with the message. If the two are identical, it ensures the contents have not been altered.

When receiving messages, return asynchronous receipts to the first enabled AS2 delivery exchange for the partner instead of the disposition notification URL from the message

Select this check box to override the disposition notification URL for receipts that your partner set in the message your community received. When selected, the partner's disposition notification URL in the MIME header of the inbound message is ignored, and your community sends the receipt via the first enabled AS2 delivery exchange in the partner profile.

Specify message attributes to be packaged with message

This option is available only if your license has a **true** permission for allowCustomMetadataInMessageHeaders.

Checking this enables you to include any meta-data elements you want in the MIME headers of the messages your community sends. This can serve several purposes. It lets your community forward payload-related information to a partner. It also lets a community or partner use such data to trigger message handling actions.

The meta-data that can be added are in addition to the meta-data already in MIME headers, such as sender and receiver, content type, disposition notification and so on.

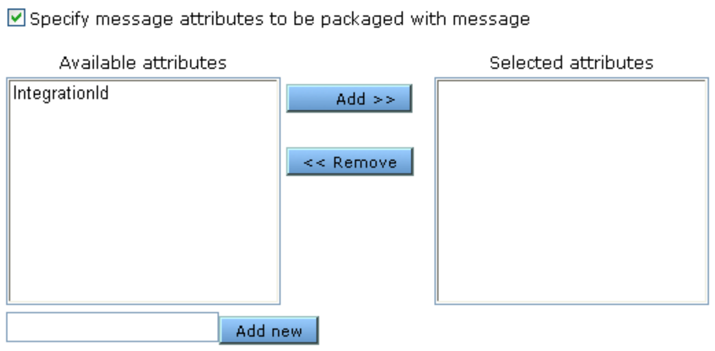


Figure 114. Available and selected attributes list boxes

There are several ways to add meta-data elements to the available attributes list on this tab. You can type an attribute name in the field below the available attributes list and click **Add new**. Or, you can go to the message handler area of the user interface and add an attribute (see the chapter on message handling). Any attributes added in the message handler area appear on the available attributes list on this tab.

We recommend these guidelines for attribute names: Make attribute names a single text string. For names that use multiple words, do not use spaces between the words. Use camel case for names that include two or more words (for example, AttributeName). Do not use non-alphanumeric characters in attribute names (for example, commas, periods, hyphens, asterisks, ampersands and so on).

To add attributes to headers of outbound documents, use the **Add** button on this tab to move a selected attribute from the available attributes list to the selected attributes list. Click **Save changes** when done.

In addition to selecting meta-data elements to include in headers, you must separately establish the values for the attributes. There are several ways to do this, including:

Message attributes tab. Use the message attributes tab on a transport maintenance page to set attribute values by use of directory mapping or fixed values. For more information see the message attributes tab topic in the transport maintenance chapter.

Message handler. In the message handler area, you can set a fixed value for an attribute. If the payload is an XML document, the value can be parsed by specifying an XPath. For more information see the chapter on message handling.

Inline processing. If you are a licensed SDK user, you can build a Java class and use it to provide attribute values or perform other message actions.

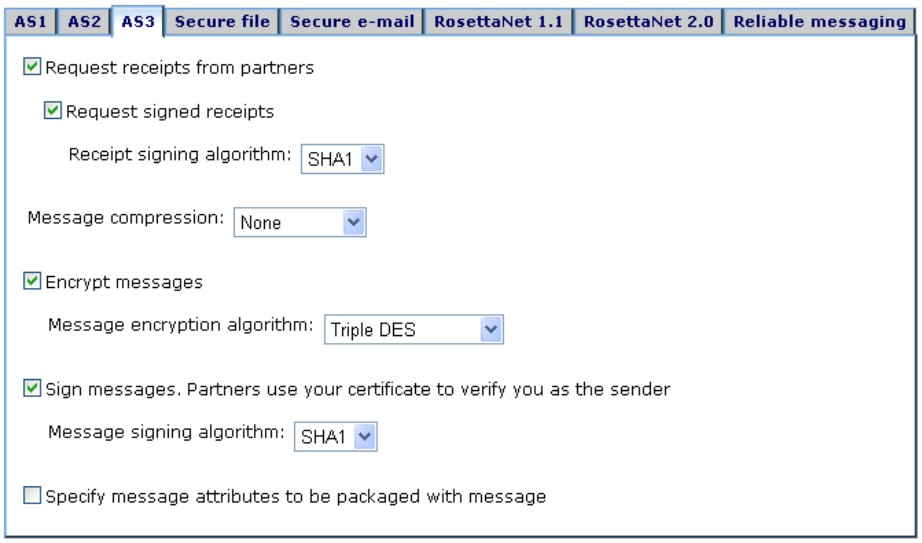
Upon receiving and unpackaging a message with extra meta-data, a partner who uses Cyclone Activator 5.3.2 or later can use the added-meta-data for post processing, inline processing or file system integration. Inform the partner of the meta-data elements you add to outbound messages.

The added meta-data in the header of the outbound document is in the following format:

X-Cyclone-Metadata-AttributeName: Value

AS3 default settings

The AS3 tab shows default outbound message settings for the AS3 message protocol. The settings affect all outbound documents for all communities, unless superseded by community- or partner-level settings. You can change these as you need. Figure 115 displays the default AS3 settings tab.



☑ Request receipts from partners

☑ Request signed receipts

Receipt signing algorithm: SHA1

Message compression: None

☑ Encrypt messages

Message encryption algorithm: Triple DES

☑ Sign messages. Partners use your certificate to verify you as the sender

Message signing algorithm: SHA1

☐ Specify message attributes to be packaged with message

Figure 115. Default AS3 settings

Request receipts from partners

Select this check box to have your partners send receipts to you upon receiving your messages. The receipts are signed or unsigned, depending on your selection in the request signed receipts check box.

Request signed receipts

Select this check box to have your partners sign the receipts they send you.

Receipt signing algorithm

This is the algorithm you want partners' trading engines to use when creating a hash of the unencrypted receipt. This hash is a number that is encrypted with the partner's private key. It is decrypted by the community using the partner's public key. The community rehashes the decrypted receipt and compares the result with the hash that came with the receipt. If the two are identical, it ensures the contents have not been altered.

Message compression

Select a method for compressing the messages you send. This is optional.

Compressing data before sending increases throughput. You might want to consult with your partner about compressing and uncompressing as pre- and post-processing steps. You and your partner can compress or not independently, as long as the receiver's system supports uncompressing.

The compression options are:

None. Select this if you do not want compression. Also, you must select this to exchange messages with partners whose interoperable software cannot uncompress documents.

EDIINT/ZLIB is for trading with partners who use version 4.2 or later of Cyclone Activator or who use an EDIINT-compliant trading engine other than Cyclone Activator.

MIME/GZIP is for trading with partners who use versions of Cyclone Activator earlier than 4.2. You also can select this if your partner uses a trading engine other than Cyclone Activator that supports GZIP compression.

Encrypt messages

Select this check box to encrypt the messages you send.

Message encryption algorithm

If you select encrypt messages, select an algorithm.

Sign messages. Partners use your certificate to verify you as the sender

Select this check box to digitally sign the messages you send.

Message signing algorithm

The algorithm for creating a hash of the unencrypted message. This hash is a number that is encrypted with your community’s private key. It is decrypted by the partner using the community’s public key. The partner rehashes the decrypted message and compares the result with the hash that came with the message. If the two are identical, it ensures the contents have not been altered.

Specify message attributes to be packaged with message

This option is available only if your license has a **true** permission for allowCustomMetadataInMessageHeaders.

Checking this enables you to include any meta-data elements you want in the MIME headers of the messages your community sends. This can serve several purposes. It lets your community forward payload-related information to a partner. It also lets a community or partner use such data to trigger message handling actions.

The meta-data that can be added are in addition to the meta-data already in MIME headers, such as sender and receiver, content type, disposition notification and so on.

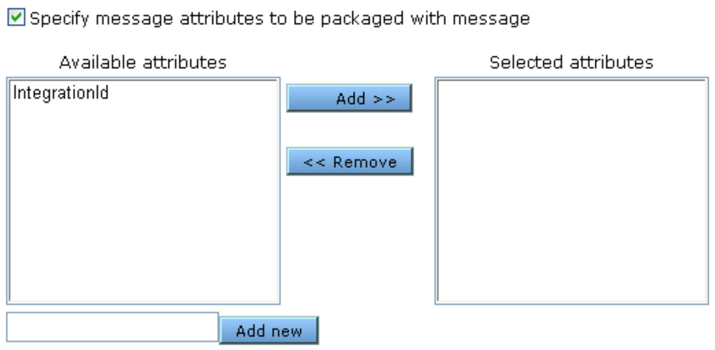


Figure 116. Available and selected attributes list boxes

There are several ways to add meta-data elements to the available attributes list on this tab. You can type an attribute name in the field below the available attributes list and click **Add new**. Or, you can go to the message handler area of the user interface and add an attribute (see the chapter on message handling). Any attributes added in the message handler area appear on the available attributes list on this tab.

We recommend these guidelines for attribute names: Make attribute names a single text string. For names that use multiple words, do not use spaces between the words. Use camel case for names that include two or more words (for example, AttributeName). Do not use non-alphanumeric characters in attribute names (for example, commas, periods, hyphens, asterisks, ampersands and so on).

To add attributes to headers of outbound documents, use the **Add** button on this tab to move a selected attribute from the available attributes list to the selected attributes list. Click **Save changes** when done.

In addition to selecting meta-data elements to include in headers, you must separately establish the values for the attributes. There are several ways to do this, including:

Message attributes tab. Use the message attributes tab on a transport maintenance page to set attribute values by use of directory mapping or fixed values. For more information see the message attributes tab topic in the transport maintenance chapter.

Message handler. In the message handler area, you can set a fixed value for an attribute. If the payload is an XML document, the value can be parsed by specifying an XPath. For more information see the chapter on message handling.

Inline processing. If you are a licensed SDK user, you can build a Java class and use it to provide attribute values or perform other message actions.

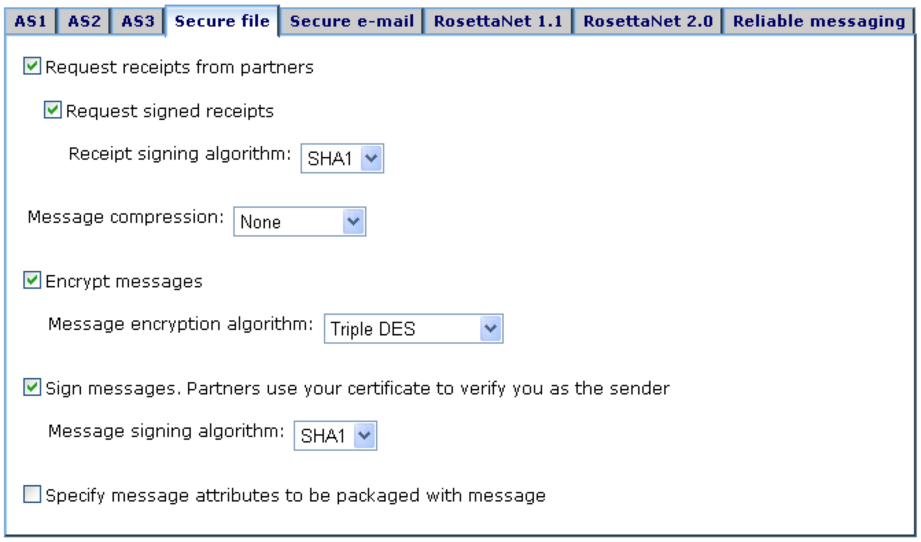
Upon receiving and unpackaging a message with extra meta-data, a partner who uses Cyclone Activator 5.3.2 or later can use the added-meta-data for post processing, inline processing or file system integration. Inform the partner of the meta-data elements you add to outbound messages.

The added meta-data in the header of the outbound document is in the following format:

X-Cyclone-Metadata-AttributeName: Value

Secure file default settings

The secure file tab shows default outbound message settings for the secure file message protocol. The settings affect all outbound documents for all communities, unless superseded by community- or partner-level settings. You can change these as you need. Figure 117 displays the default secure file settings tab.



☑ Request receipts from partners

☑ Request signed receipts

Receipt signing algorithm: SHA1

Message compression: None

☑ Encrypt messages

Message encryption algorithm: Triple DES

☑ Sign messages. Partners use your certificate to verify you as the sender

Message signing algorithm: SHA1

☐ Specify message attributes to be packaged with message

Figure 117. Default secure file settings

Request receipts from partners

Select this check box to have your partners send receipts to you upon receiving your messages. The receipts are signed or unsigned, depending on your selection in the request signed receipts check box.

Request signed receipts

Select this check box to have your partners sign the receipts they send you.

Receipt signing algorithm

This is the algorithm you want partners' trading engines to use when creating a hash of the unencrypted receipt. This hash is a number that is encrypted with the partner's private key. It is decrypted by the community using the partner's public key. The community rehashes the decrypted receipt and compares the result with the hash that came with the receipt. If the two are identical, it ensures the contents have not been altered.

Message compression

Select a method for compressing the messages you send. This is optional.

Compressing data before sending increases throughput. You might want to consult with your partner about compressing and uncompressing as pre- and post-processing steps. You and your partner can compress or not independently, as long as the receiver's system supports uncompressing.

The compression options are:

None. Select this if you do not want compression. Also, you must select this to exchange messages with partners whose interoperable software cannot uncompress documents.

EDIINT/ZLIB is for trading with partners who use version 4.2 or later of Cyclone Activator or who use an EDIINT-compliant trading engine other than Cyclone Activator.

MIME/GZIP is for trading with partners who use versions of Cyclone Activator earlier than 4.2. You also can select this if your partner uses a trading engine other than Cyclone Activator that supports GZIP compression.

Encrypt messages

Select this check box to encrypt the messages you send.

Message encryption algorithm

If you select encrypt messages, select an algorithm.

Sign messages. Partners use your certificate to verify you as the sender

Select this check box to digitally sign the messages you send.

Message signing algorithm

The algorithm for creating a hash of the unencrypted message. This hash is a number that is encrypted with your community's private key. It is decrypted by the partner using the community's public key. The partner rehashes the decrypted message and compares the result with the hash that came with the message. If the two are identical, it ensures the contents have not been altered.

Specify message attributes to be packaged with message

This option is available only if your license has a **true** permission for allowCustomMetadataInMessageHeaders.

Checking this enables you to include any meta-data elements you want in the MIME headers of the messages your community sends. This can serve several purposes. It lets your community forward payload-related information to a partner. It also lets a community or partner use such data to trigger message handling actions.

The meta-data that can be added are in addition to the meta-data already in MIME headers, such as sender and receiver, content type, disposition notification and so on.

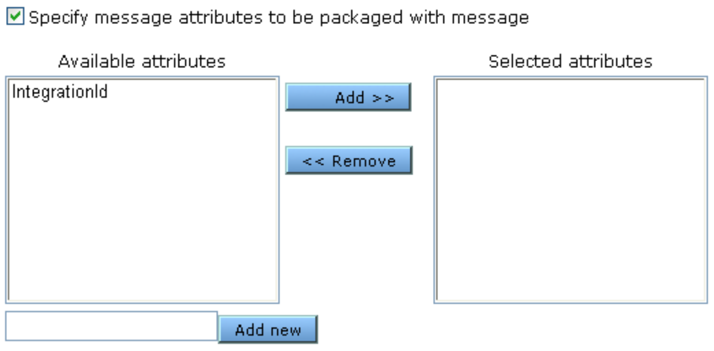


Figure 118. Available and selected attributes list boxes

There are several ways to add meta-data elements to the available attributes list on this tab. You can type an attribute name in the field below the available attributes list and click **Add new**. Or, you can go to the message handler area of the user interface and add an attribute (see the chapter on message handling). Any attributes added in the message handler area appear on the available attributes list on this tab.

We recommend these guidelines for attribute names: Make attribute names a single text string. For names that use multiple words, do not use spaces between the words. Use camel case for names that include two or more words (for example, AttributeName). Do not use non-alphanumeric characters in attribute names (for example, commas, periods, hyphens, asterisks, ampersands and so on).

To add attributes to headers of outbound documents, use the **Add** button on this tab to move a selected attribute from the available attributes list to the selected attributes list. Click **Save changes** when done.

In addition to selecting meta-data elements to include in headers, you must separately establish the values for the attributes. There are several ways to do this, including:

Message attributes tab. Use the message attributes tab on a transport maintenance page to set attribute values by use of directory mapping or fixed values. For more information see the message attributes tab topic in the transport maintenance chapter.

Message handler. In the message handler area, you can set a fixed value for an attribute. If the payload is an XML document, the value can be parsed by specifying an XPath. For more information see the chapter on message handling.

Inline processing. If you are a licensed SDK user, you can build a Java class and use it to provide attribute values or perform other message actions.

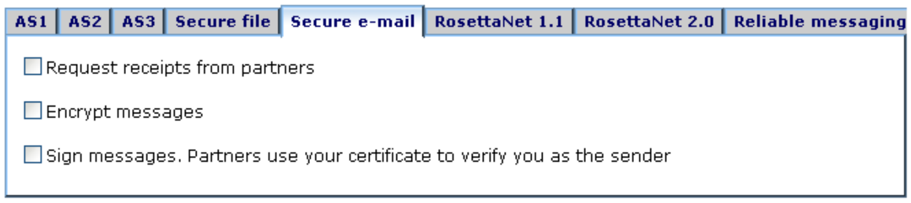
Upon receiving and unpackaging a message with extra meta-data, a partner who uses Cyclone Activator 5.3.2 or later can use the added-meta-data for post processing, inline processing or file system integration. Inform the partner of the meta-data elements you add to outbound messages.

The added meta-data in the header of the outbound document is in the following format:

X-Cyclone-Metadata-AttributeName: Value

Secure e-mail default settings

The secure e-mail tab shows default outbound message settings for the secure e-mail message protocol. The settings affect all outbound documents for all communities, unless superseded by community- or partner-level settings. You can change these as you need. Figure 119 displays the default secure e-mail settings tab.



☐ Request receipts from partners

☐ Encrypt messages

☐ Sign messages. Partners use your certificate to verify you as the sender

Figure 119. Default secure e-mail settings

Request receipts from partners

Select this check box to have your partners send receipts to you upon receiving your messages. The receipts are signed or unsigned, depending on your selection in the request signed receipts check box.

Encrypt messages

Select this check box to encrypt the messages you send.

Message encryption algorithm

If you select encrypt messages, select an algorithm.

Sign messages. Partners use your certificate to verify you as the sender

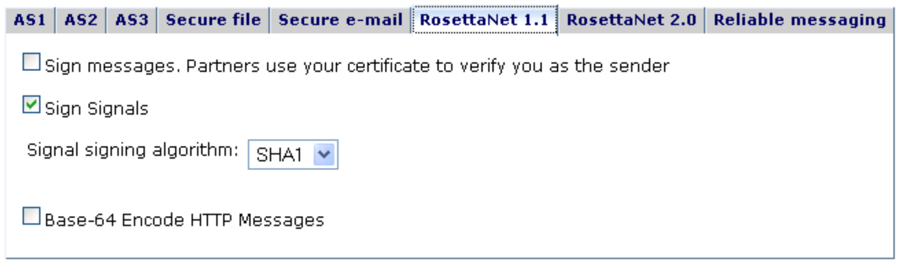
Select this check box to digitally sign the messages you send.

Message signing algorithm

The algorithm for creating a hash of the unencrypted message. This hash is a number that is encrypted with your community’s private key. It is decrypted by the partner using the community’s public key. The partner rehashes the decrypted message and compares the result with the hash that came with the message. If the two are identical, it ensures the contents have not been altered.

RosettaNet 1.1 default settings

The RosettaNet 1.1 tab shows default outbound message settings for the RosettaNet 1.1 message protocol. The settings affect all outbound documents for all communities, unless superseded by community- or partner-level settings. You can change these as you need. Figure 120 displays the default AS3 settings tab.



☐ Sign messages. Partners use your certificate to verify you as the sender

☑ Sign Signals

Signal signing algorithm: SHA1

☐ Base-64 Encode HTTP Messages

Figure 120. Default RosettaNet 1.1 settings

Sign messages. Partners use your certificate to verify you as the sender

Select this check box to digitally sign the messages you send.

Message signing algorithm

The algorithm for creating a hash of the unencrypted message. This hash is a number that is encrypted with your community’s private key. It is decrypted by the partner using the community’s public key. The partner rehashes the decrypted message and compares the result with the hash that came with the message. If the two are identical, it ensures the contents have not been altered.

Sign Signals

Sign the signals you send to partners.

Signal signing algorithm

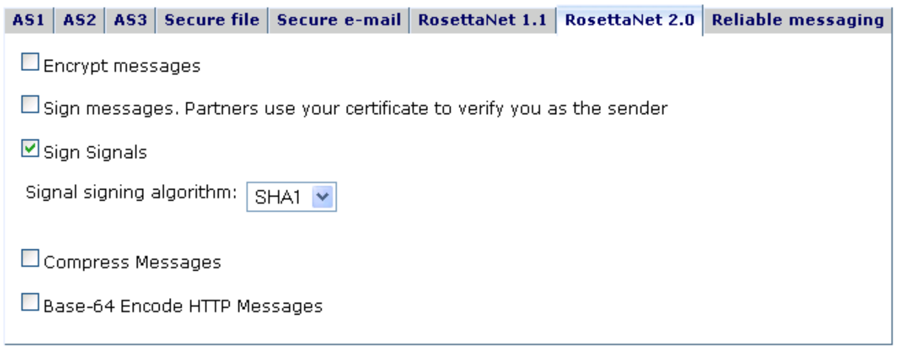
If you choose to sign signals, the signing algorithm to use.

Base-64 Encode HTTP Messages

Convert outbound messages to base 64. If not selected, messages are sent in binary format.

RosettaNet 2.0 default settings

The RosettaNet 2.0 tab shows default outbound message settings for the RosettaNet 2.0 message protocol. The settings affect all outbound documents for all communities, unless superseded by community- or partner-level settings. You can change these as you need. Figure 121 displays the default RosettaNet 2.0 settings tab.



☐ Encrypt messages

☐ Sign messages. Partners use your certificate to verify you as the sender

☑ Sign Signals

Signal signing algorithm: SHA1

☐ Compress Messages

☐ Base-64 Encode HTTP Messages

Figure 121. Default RosettaNet 2.0 settings

Encrypt messages

Select this check box to encrypt the messages you send.

Message encryption algorithm

If you select encrypt messages, select an algorithm.

Encrypt service content and attachments

If encrypt messages is also selected, the payload is encrypted (service-content and attachments).

Sign messages. Partners use your certificate to verify you as the sender

Select this check box to digitally sign the messages you send.

Message signing algorithm

The algorithm for creating a hash of the unencrypted message. This hash is a number that is encrypted with your community's private key. It is decrypted by the partner using the community's public key. The partner rehashes the decrypted message and compares the result with the hash that came with the message. If the two are identical, it ensures the contents have not been altered.

Sign Signals

Sign the signals you send to partners.

Signal signing algorithm

If you choose to sign signals, the signing algorithm to use.

Compress Messages

Compress outbound messages.

Base-64 Encode HTTP Messages

Convert outbound messages to base 64. If not selected, messages are sent in binary format.

Reliable messaging default settings

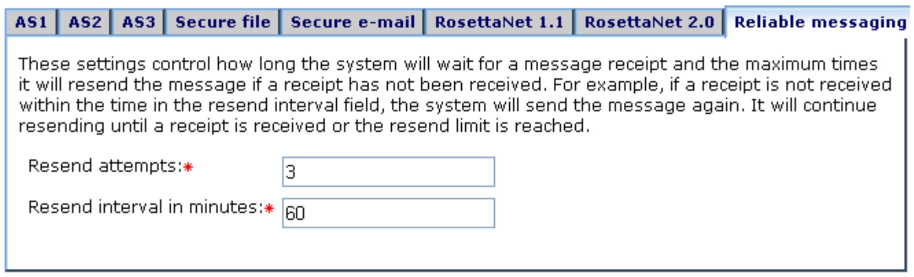
The reliable messaging tab shows default outbound settings for message resend attempts. The tab applies only to messages for which receipts are expected. The settings affect all outbound documents for all communities, unless superseded by community- or partner-level settings. You can change these as you need. Figure 122 displays the default reliable messaging tab.

The tab controls how long the system will wait for a message receipt and the maximum times it will resend the message if a receipt has not been received.

For example, if a receipt is not received within the time in the resend interval field, the system will send the message again. It will continue resending until a receipt is received or the resend limit is reached.

If the resend limit is reached and a receipt has not been received, the message is given a failed status. You can search for and manually resubmit failed messages in Message tracker. See Resubmitting messages on page 415.

This tab only applies to successfully sent messages for which receipts are expected. It does not apply to messages that fail to send due to a transport problem. For information about resending due to transport issues, see information about the Retries field on the Advanced tab of a transport maintenance page (Transport maintenance on page 247).



These settings control how long the system will wait for a message receipt and the maximum times it will resend the message if a receipt has not been received. For example, if a receipt is not received within the time in the resend interval field, the system will send the message again. It will continue resending until a receipt is received or the resend limit is reached.

Resend attempts:* 3

Resend interval in minutes:* 60

Figure 122. Default reliable messaging settings

Resend attempts

The maximum number of times the trading engine should resend a message if a receipt has not been received.

Resend interval in minutes

The time the system waits for a receipt before resending the message.

Community collaboration settings

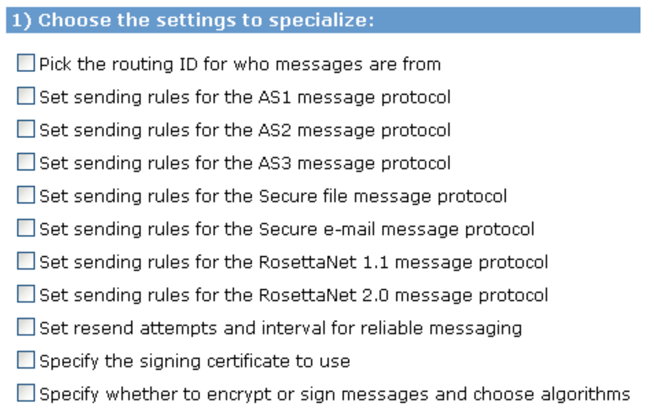
You can specify collaboration settings that apply only to one community. This allows you to define exceptions to the default outbound message settings.

For example, the default collaboration settings could specify that messages sent over the AS2 protocol should return synchronous acknowledgements. Community A might want asynchronous acknowledgements, or perhaps none at all, for the AS2 protocol. Community A can specify his AS2 collaboration settings according to his needs, while using the default collaboration settings for sending message via other protocols.

You only need to specify exceptions to the default settings. For any settings you do not change at the community level, the global defaults are used. See Default collaboration settings on page 371.

You do not need community-specific collaborations if your installation defines only one community or if all communities want to send messages the same way.

To view or change community-specific collaboration settings, click **Collaboration settings** on the navigation graphic on the community summary page. From there, you can specify the settings you want to define for the community, as shown in Figure 123.


1) Choose the settings to specialize:

- ☐ Pick the routing ID for who messages are from
- ☐ Set sending rules for the AS1 message protocol
- ☐ Set sending rules for the AS2 message protocol
- ☐ Set sending rules for the AS3 message protocol
- ☐ Set sending rules for the Secure file message protocol
- ☐ Set sending rules for the Secure e-mail message protocol
- ☐ Set sending rules for the RosettaNet 1.1 message protocol
- ☐ Set sending rules for the RosettaNet 2.0 message protocol
- ☐ Set resend attempts and interval for reliable messaging
- ☐ Specify the signing certificate to use
- ☐ Specify whether to encrypt or sign messages and choose algorithms

Figure 123. Community collaboration settings options

To specify how a community sends messages to a specific partner, see Partner collaboration settings on page 394.

The following describes the override options.

Pick the routing ID for who messages are from

A community can have multiple routing IDs. If you select this, you can choose a single routing ID to apply the community-level settings. The settings then affect only messages sent with that ID. Messages sent with other IDs use the global defaults.

If you do not select this, the community-level settings apply to messages sent using all of the community's routing IDs.

Set sending rules for the AS1 message protocol

Lets you override the AS1 global defaults. For field descriptions see AS1 default settings on page 371.

Set sending rules for the AS2 message protocol

Lets you override the AS2 global defaults. For field descriptions see AS2 default settings on page 375.

Set sending rules for the AS3 message protocol

Lets you override the AS2 global defaults. For field descriptions see AS3 default settings on page 379.

Set sending rules for the Secure file message protocol

Lets you override the secure file global defaults. For field descriptions see Secure file default settings on page 383.

Set sending rules for the Secure e-mail message protocol

Lets you override the secure e-mail global defaults. For field descriptions see Secure e-mail default settings on page 387.

Set sending rule for the RosettaNet 1.1 message protocol

Lets you override the RosettaNet 1.1 global defaults. For field descriptions see RosettaNet 1.1 default settings on page 388.

Set sending rule for the RosettaNet 2.0 message protocol

Lets you override the RosettaNet 2.0 global defaults. For field descriptions see RosettaNet 2.0 default settings on page 389.

Set resend attempts and interval for reliable messaging

Lets you override the defaults for resending messages. For field descriptions see Reliable messaging default settings on page 390.

Specify the signing certificate to use

Lets you specify the certificate to use for signing messages, if it is different than the community default signing certificate.

Specify whether to encrypt or sign messages and choose algorithms

Lets you override the global defaults for encrypting and signing messages. This option overrides not only the default collaboration settings for your installation, but the protocol-specific signing and encrypting settings you might have set up for this community.

When selected the following fields display.

Encrypt messages

Select this check box to encrypt the messages you send.

Message encryption algorithm

If you select encrypt messages, select an algorithm.

Sign messages. Partners use your certificate to verify you as the sender

Select this check box to digitally sign the messages you send.

Message signing algorithm

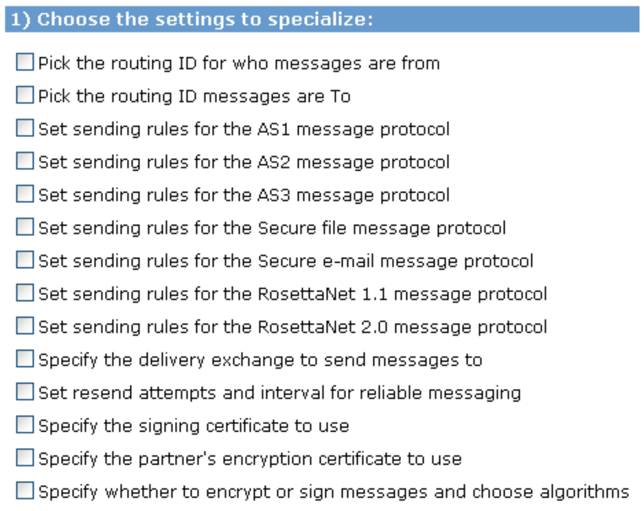
The algorithm for creating a hash of the unencrypted message. This hash is a number that is encrypted with your community's private key. It is decrypted by the partner using the community's public key. The partner rehashes the decrypted message and compares the result with the hash that came with the message. If the two are identical, it ensures the contents have not been altered.

Partner collaboration settings

You can specify collaboration settings that apply only to one partner of a community. This allows you to define exceptions to the default and community-level collaboration settings for sending messages to a single partner.

For example, your default collaboration settings could specify that messages sent over the secure file protocol should return signed acknowledgements. Partner A might want to send unsigned acknowledgements over FTP. You can specify that Partner A send unsigned acknowledgements for the secure file protocol, while all other partners in the community use the default or community-level settings.

To access partner-specific collaboration settings, click **Collaboration settings** on the navigation graphic on the community summary page. If partner-level settings have been established, click the partner's name on the left side of the page. Otherwise, click **Specialize collaboration settings for a partner** and follow the prompts to select a partner. Then specify the settings you want to define for the partner, as shown in Figure 123.


1) Choose the settings to specialize:

- ☐ Pick the routing ID for who messages are from
- ☐ Pick the routing ID messages are To
- ☐ Set sending rules for the AS1 message protocol
- ☐ Set sending rules for the AS2 message protocol
- ☐ Set sending rules for the AS3 message protocol
- ☐ Set sending rules for the Secure file message protocol
- ☐ Set sending rules for the Secure e-mail message protocol
- ☐ Set sending rules for the RosettaNet 1.1 message protocol
- ☐ Set sending rules for the RosettaNet 2.0 message protocol
- ☐ Specify the delivery exchange to send messages to
- ☐ Set resend attempts and interval for reliable messaging
- ☐ Specify the signing certificate to use
- ☐ Specify the partner's encryption certificate to use
- ☐ Specify whether to encrypt or sign messages and choose algorithms

Figure 124. Partner collaboration settings options

The following describes the override options.

Pick the routing ID for who messages are from

A community can have multiple routing IDs. If you select this, you can choose a single routing ID to apply the partner-level settings. All messages from the community to this partner use this community routing ID.

If you do not select this, the partner-level settings apply to messages sent using all of the community's routing IDs.

Pick the routing ID messages are to

A partner can have multiple routing IDs. If you select this, you can choose a single routing ID to apply the partner-level settings. All messages from the community to this partner use this partner routing ID.

If you do not select this, the partner-level settings apply to messages sent using all of the partner's routing IDs.

Set sending rules for the AS1 message protocol

Lets you override the AS1 global defaults. For field descriptions see AS1 default settings on page 371.

Set sending rules for the AS2 message protocol

Lets you override the AS2 global defaults. For field descriptions see AS2 default settings on page 375.

Set sending rules for the AS3 message protocol

Lets you override the AS2 global defaults. For field descriptions see AS3 default settings on page 379.

Set sending rules for the Secure file message protocol

Lets you override the secure file global defaults. For field descriptions see Secure file default settings on page 383.

Set sending rules for the Secure e-mail message protocol

Lets you override the secure e-mail global defaults. For field descriptions see Secure e-mail default settings on page 387.

Set sending rule for the RosettaNet 1.1 message protocol

Lets you override the RosettaNet 1.1 global defaults. For field descriptions see RosettaNet 1.1 default settings on page 388.

Set sending rule for the RosettaNet 2.0 message protocol

Lets you override the RosettaNet 2.0 global defaults. For field descriptions see RosettaNet 2.0 default settings on page 389.

Specify the delivery exchange to send messages to

Lets you choose the transport for sending messages to this partner.

Set resend attempts and interval for reliable messaging

Lets you override the defaults for resending messages. For field descriptions see Reliable messaging default settings on page 390.

Specify the signing certificate to use

Lets you specify the certificate to use for signing messages, if it is different than the community default signing certificate.

Specify the partner’s encryption certificate to use

Lets you choose which of the partner’s certificates to use to encrypt messages.

Specify whether to encrypt or sign messages and choose algorithms

Lets you override the global and community-level defaults for encrypting and signing messages. This option overrides not only the default collaboration settings for your installation, but the protocol-specific signing and encrypting settings you might have set up for this partner.

When selected the following fields display.

Encrypt messages

Select this check box to encrypt the messages you send.

Message encryption algorithm

If you select encrypt messages, select an algorithm.

Sign messages. Partners use your certificate to verify you as the sender

Select this check box to digitally sign the messages you send.

Message signing algorithm

The algorithm for creating a hash of the unencrypted message. This hash is a number that is encrypted with your community's private key. It is decrypted by the partner using the community's public key. The partner rehashes the decrypted message and compares the result with the hash that came with the message. If the two are identical, it ensures the contents have not been altered.

❖ ❖ ❖

20 Inbound message validation

The message validation rules page in the user interface is for specifying whether a community accepts or rejects messages from partners. You can specify whether a community accepts or rejects:

- Duplicate EDI documents
- Signed or unsigned messages
- Encrypted or plain text messages

For information about packaging of outbound messages, see Collaboration settings on page 367.

Concepts

- Opening the validation page
- Duplicate EDI documents
- Signed or unsigned messages on page 401
- Encrypted or plain text messages on page 402
- Duplicate CSOS orders on page 403

Opening the validation page

To access the page that controls whether to accept or reject documents from partners, click the **Message validation** icon on the navigation graphic at the top of the community summary page. The “configure message validation rules” page displays.

Duplicate EDI documents

You control whether a community accepts or rejects duplicate EDI documents on the duplicate messages tab of the message validation rules page.

By default, the radio button for rejecting duplicate EDI messages from partners is selected on the tab. When selected, the trading engine checks whether an inbound message duplicates a previously received message in the following ways:

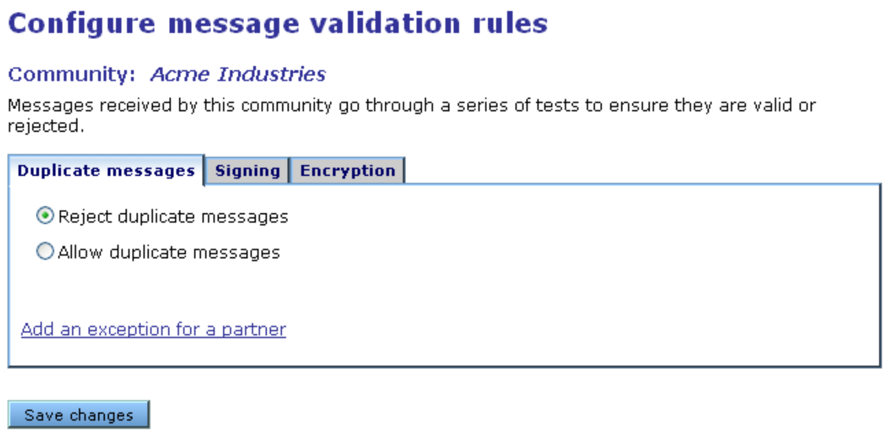
- Duplicate control number

- Duplicate sender name
- Duplicate sender routing ID
- Duplicate receiver name
- Duplicate receiver routing ID

If all of these values are the same as those in a previously received message, the message is given a failed status. You can search for failed messages in Message tracker.

The trading engine only checks for duplicates of EDI documents. Specifically, this means documents with the following MIME types:

- application/EDI-X12 (X12 document)
- application/EDIFACT (EDIFACT document)
- application/EDI-consent (TRADACOMS document)



Configure message validation rules

Community: *Acme Industries*

Messages received by this community go through a series of tests to ensure they are valid or rejected.

[✓ Reject duplicate messages]

Allow duplicate messages

Figure 125. Duplicate messages tab

In the event a batch parent document is received and is split into multiple child documents, all with the same control number as the parent, the trading engine does not consider siblings of the same parent to be duplicates. This is a processing exception to accommodate custom EDI splitters that can be used in place of the system’s standard EDI splitters.

Aside from choosing whether a community can allow or reject duplicates, you can set up exceptions to the selected behavior on a per-partner basis.

Whether you choose to reject or allow duplicate EDI messages, the choice applies to all EDI messages received for the community, unless you define partner-specific exceptions.

To add a partner-specific exception to the selected behavior, click **Add an exception for a partner**. A wizard displays to help you locate and select the partner or partners you wish to add. You can add multiple partners to the exception list at one time using the wizard.

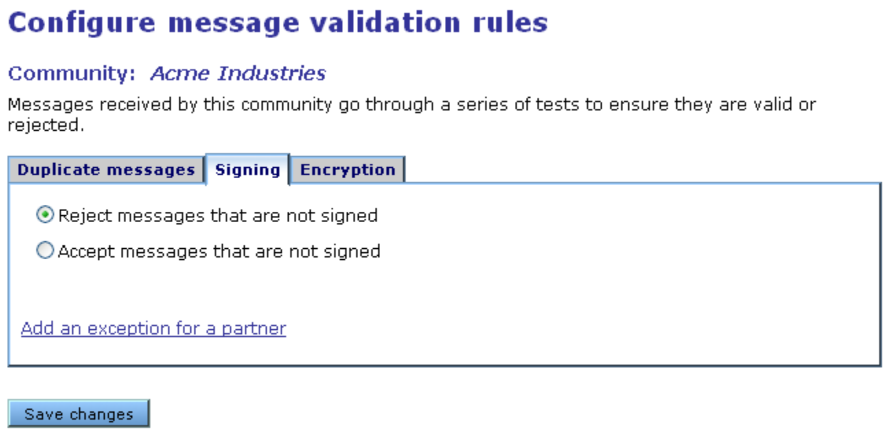
The trading engine applies the opposite of the selected behavior to any partners that display in the exception list.

Signed or unsigned messages

You control whether a community accepts or rejects unsigned messages on the signing tab of the message validation rules page.

The default selection is to reject messages that are not signed. This means the messages a partner sends must be signed with your partner's private encryption key. Your community verifies the signature with the public key in the digital certificate associated with the partner profile. To receive signed messages, the partner's certificate and public key must be included in the partner profile.

The other option is to accept unsigned messages. If you select this, not only are unsigned messages accepted, but also signed messages if the partner's certificate is in the partner profile.


Configure message validation rules

Community: *Acme Industries*

Messages received by this community go through a series of tests to ensure they are valid or rejected.

[✓ Reject messages that are not signed]

Accept messages that are not signed

Figure 126. Signing tab

Aside from choosing whether a community can accept or reject unsigned messages, you can set up exceptions to the selected behavior on a per-partner basis.

Whether you choose to reject or accept unsigned messages, the choice applies to all messages received for the community, unless you define partner-specific exceptions. However, keep in mind that if you choose to accepted unsigned messages, signed messages still will be accepted, provided the proper certificate is in place.

To add a partner-specific exception to the selected behavior, click **Add an exception for a partner**. A wizard displays to help you locate and select the partner or partners you wish to add. You can add multiple partners to the exception list at one time using the wizard.

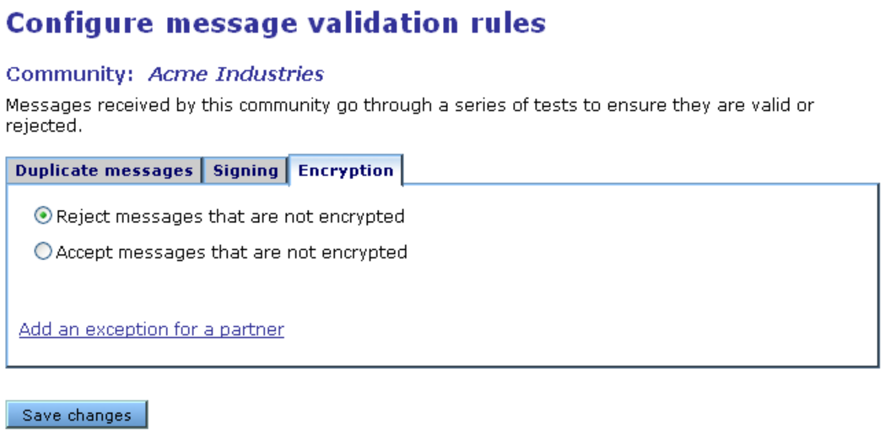
The trading engine applies the opposite of the selected behavior to any partners that display in the exception list.

Encrypted or plain text messages

You control whether a community accepts or rejects plain text messages on the encryption tab of the message validation rules page.

The default selection is to reject messages that are not encrypted. This means the messages a partner sends must be encrypted with your public encryption key. Your community decrypts the message with the private key in the digital certificate associated with your community profile. To send encrypted messages, the partner must have your certificate and public key.

The other option is to accept messages that have not been encrypted. If you select this, not only are plain text messages accepted, but also encrypted messages if a private key is associated with the community profile.


Configure message validation rules

Community: *Acme Industries*

Messages received by this community go through a series of tests to ensure they are valid or rejected.

[✓ Reject messages that are not encrypted]

Accept messages that are not encrypted

Figure 127. Encryption tab

Aside from choosing whether a community can accept or reject plain text messages, you can set up exceptions to the selected behavior on a per-partner basis.

Whether you choose to reject or accept plain text messages, the choice applies to all messages received for the community, unless you define partner-specific exceptions.

To add a partner-specific exception to the selected behavior, click **Add an exception for a partner**. A wizard displays to help you locate and select the partner or partners you wish to add. You can add multiple partners to the exception list at one time using the wizard.

The trading engine applies the opposite of the selected behavior to any partners that display in the exception list.

Duplicate CSOS orders

If your user license provides authority for CSOS functionality, the CSOS duplicate orders tab displays. Otherwise, this tab is not available and you can ignore the following information.

CSOS duplicate checking is done for CSOS purchase orders your community receives from partners. If the community is a WebTrader sponsor, documents received from its WebTrader partners also are checked. Duplicate checking is not done for orders picked up from integration, unless the community that picks up the order is the named receiver (a rare case).

If your community only sends signed orders but does not receive any, duplicate checking is not applicable and you can ignore this tab.

The target documents are the EDI or XML documents defined on the document types tab of the identify CSOS purchase orders page.

By default, the radio button for rejecting duplicate purchase orders is selected on the CSOS duplicate orders tab. When selected, the trading engine checks whether an order duplicates a previously received order in the following ways:

- Duplicate order number
- Duplicate DEA registration number
- Duplicate sender name
- Duplicate receiver name

If all of these values are the same as those in a previously received order, the document is given a failed status. You can search for failed documents in Message tracker.

If you also have configured the trading engine to check for duplicate EDI documents, checking for duplicate CSOS orders is performed in addition to the duplicate EDI checking.

Aside from choosing whether a community can allow or reject duplicates, you can set up exceptions to the selected behavior on a per-partner basis.

Whether you choose to reject or allow duplicate orders, the choice applies to all orders received for the community, unless you define partner-specific exceptions.

To add a partner-specific exception to the selected behavior, click **Add an exception for a partner**. A wizard displays to help you locate and select the partner or partners you wish to add. You can add multiple partners to the exception list at one time using the wizard.

The trading engine applies the opposite of the selected behavior to any partners that display in the exception list.

Note that the CSOS duplicate order tabs can be found on two pages in the user interface. One location is the identify CSOS purchase orders page at **CSOS > Configure CSOS**. The other is the message validation rules page, which is opened by clicking **Message validation** on the navigation graphic at the top of a community summary page. To configure CSOS duplicate checking, you only need to use one of these pages.

❖ ❖ ❖

21 Message handling

Message handling refers to optional processing of inbound or outbound messages based on meta-data attributes and actions. The trading engine lets you set up conditions to:

- Copy messages to parties other than the sending or receiving party
- Reject messages
- Perform custom processing using your own Java class

Message handling involves performing message actions. Message actions are triggered by single or multiple conditions, which are a combination of attributes and operators. For example, you can order that whenever a community sends message to partner A, a copy is sent to partner B.

Message actions can be applied to inbound and outbound messages. For inbound messages, message actions are applied after a message has been validated, unpackaged and parsed, but before the payload is sent to a back-end system via an integration transport. For outbound messages, message actions are applied after a document has been picked up from integration, but before it has been packaged.

Concepts

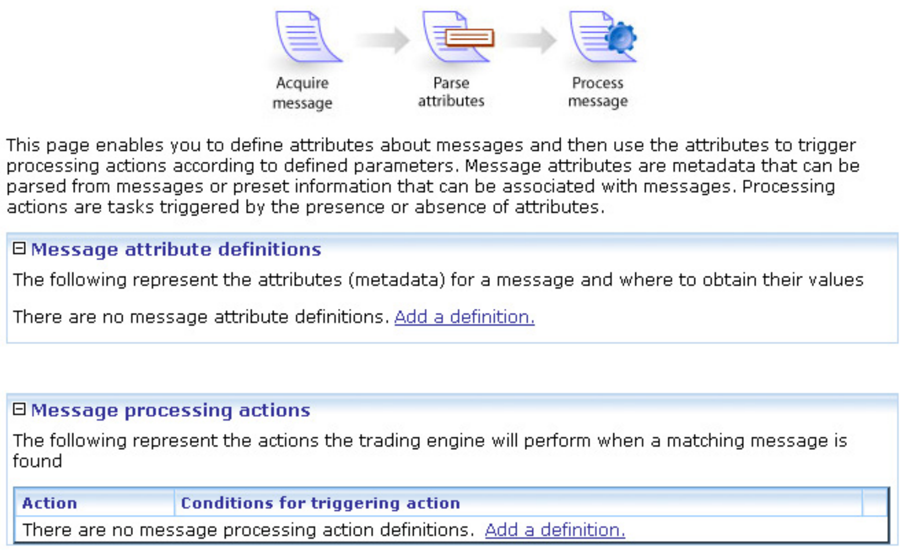
- Setting up message actions on page 405

Procedure

- Define message attributes on page 408
- Define message actions on page 409

Setting up message actions

To reach the area of the user interface for setting up message actions, click **Message handler** on the navigation graphic at the top of the community summary page. Message actions can be customized to a fine degree. We recommend that you explore the options to become familiar with them and design the actions that you want.



This page enables you to define attributes about messages and then use the attributes to trigger processing actions according to defined parameters. Message attributes are metadata that can be parsed from messages or preset information that can be associated with messages. Processing actions are tasks triggered by the presence or absence of attributes.

⊟ Message attribute definitions

The following represent the attributes (metadata) for a message and where to obtain their values

There are no message attribute definitions. Add a definition.

⊟ Message processing actions

The following represent the actions the trading engine will perform when a matching message is found

Action	Conditions for triggering action
There are no message processing action definitions. Add a definition.	

Figure 128. Message action page

Setting up message actions can be a two-step process, as reflected by the two areas on the message handling page. The first step, defining attributes, is optional unless the attribute you need is not one of the already defined default attributes. The second step involves using attributes and values to set up conditions for triggering message actions.

1 **Define attributes**. Attributes are name-value pairs whose values are extracted from messages through parsing or have fixed values. The trading engine can parse only XML messages, but attributes with fixed values can apply to EDI, XML or binary documents.

For example, you can have the system parse certain XML messages for values of an attribute named **POAmount**. Or, you can instruct the system to apply an attribute named **SupplyChain** with a fixed value of **Retail** to certain messages.

You can define attributes to be active only when certain conditions occur. This lets you customize attributes to a high degree of specificity.

Defining attributes is optional. Attributes you define are in addition to default attributes. The system knows the names and values of the following attributes for all document types. You do not have to define them.

Consumption file name

The name of the message picked up from an integration or delivery transport.

Production file name

The name of the message sent to integration or a partner.

Sender routing ID

The routing ID of the sending party.

Receiver routing ID

The routing ID of the receiving party.

From

The name of the sending party.

To

The name of the receiving party.

Document class

The document class of the message payload (for example, X12, XML).

Content MIME type

The MIME type of the message payload. The following are commonly used types.

application/EDI-consent	Tradacoms messages
application/EDIFACT	EDIFACT messages
application/EDI-X12	X12 messages
application/octet-stream	Binary messages
application/xml	XML messages

For information about other MIME types see http://www.mhonarc.org/~ehood/MIME/MIME.html.

Document type

For EDI documents this is the business document type, such as 894 (invoice) or 850 (purchase order). For XML documents, the document type has to be parsed from the document.

Business protocol

The message protocol for transporting the message. For example, AS1, AS2, AS3, ebXML.

EDI control ID

For EDI documents, the control ID.

Is Receipt

A boolean indicating whether the message is a receipt acknowledging a message has been received. Receipts can trigger message actions unless you set up a condition excluding them.

2 **Define actions**. From the pool of available attributes, you select an attribute and also an operator and a value. This serves as the trigger of the action. For example, you could specify an action triggers when:

Sender ID	**equals**	**Community A**
(attribute)	(operator)	(value)

You can set up multiple conditions, all of which must be met for the action to trigger.

Define message attributes

Use this procedure to define message attributes. Attributes are used for setting up conditions for message actions. See Define message actions on page 409.

A number of default attributes already have been set up. For descriptions of the default attributes see step 1 of Setting up message actions on page 405.

Steps

1 Click **Message handler** on the navigation graphic at the top of the community summary page to open the message actions page.

2 Click **Add a message attribute definition**.

3 Type a name for the attribute you are adding. Click **Add** and follow the prompts.

We recommend these guidelines for attribute names: Make attribute names a single text string. For names that use multiple words, do not use spaces between the words. Use camel case for names that include two or more words (for example, AttributeName). Avoid using non-alphanumeric characters in attribute names (for example, commas, periods, asterisks, ampersands and so on).

Another suggestion is to include a prefix for the attribute name to make it easy to identify as a user-defined attribute. For example, you could use a company name as the prefix (Company-AttributeName).

4. Define a value for the attribute, whether by parsing an XML document or setting a fixed value.

5. Set up one or more conditions for the attribute. Although optional, without at least one condition the system will try to parse all messages or apply the fixed value.

Define message actions

Use this procedure to define a message action. Before doing so, you might want to define an attribute to use for setting up a condition for a message action. See Define message attributes on page 408.

By default, message actions apply to all messages, including messages containing payloads or receipts. To exclude receipts from a message action, add a condition that sets **Is Receipt** to **false**.

Steps

1. Click **Message handler** on the navigation graphic at the top of the community summary page to open the message actions page.

2. Click **Add a message processing action**.

3. Choose an attribute for the condition and click **Next**. For descriptions of the default attributes see step 1 of Setting up message actions on page 405.

4. Specify an operator and value for triggering the action and click **Next**.

5. Select the action to perform, complete the applicable fields and click **Finish**. The following describes the available actions.

   Send a copy of the message to

   Lets you specify one or more parties to send a copy of the message.

A copy can be sent only in the same direction as the original. For example, if a community sends a message to partner A, a copy can be sent to partner B. However, after receiving a message from partner A, a community cannot send a copy to partner B.

Reject the message

Lets you reject messages that meet the specified conditions. You can type the reason for the rejection.

Perform inline processing via a Java class

The extensible architecture of the trading engine allows system integrators to apply custom logic to in-process messages as an integral part of the processing pipeline. The custom processing logic, implemented as a user-defined Java class, can be selectively applied at runtime to inbound or outbound messages as determined by message actions.

Use of this feature requires obtaining an optional developer's license.

❖ ❖ ❖

22 Message tracker

Message tracker enables you to monitor database records of traded messages. You can search for and view payloads and receipts. Click the Message tracker icon on the top toolbar to go to this area of the user interface.

Message tracker uses database records and files stored in the backup directory. To get full use of Message tracker, you must enable document backups. Backing up is the default behavior when delivery exchanges are set up. See Data backups and deletes on page 421.

Concepts

Message tracker search controls

Controls are available in the **Trading information** area on the left side of the main Message tracker page for searching for messages by:

- Sender or receiver (from or to)

- Status (any, delivered, negative response, failed, in process, waiting for receipt, resubmitted, split, ignored). If you search for **any** status, all states but **ignored** are included. The **ignored** status is applied to messages the trading engine has determined lack worthwhile content (for example, an extraneous message received in addition to a message receipt).

- Direction (any, inbound, outbound, internal, external). If you search for **any** direction, all messages are included in the search regardless of origination. A search for **inbound** finds messages received from partners and routed to integration. A search for **outbound** finds messages the trading engine picked up from integration, packaged and sent to partners.

If you are a community sponsor with WebTrader partners, a search for **internal** may find messages awaiting action by a web trader. A search for **external** does not yield useful results.

- Document ID
- Document type. This is EDI document types such as 850, 862. For ebXML, you can use values such as MessageError, Ping, Pong, StatusRequest, StatusResponse, SOAP Fault. For RosettaNet, values include Signal and Action.
- Document class. Values include Tradacoms, X12, XML, Edifact, Binary. For other classes, you can use the content MIME type without the application\ prefix (for example, use octet-stream instead of application/octet-stream).
- MIME type. The content MIME type of a document. For example:

application/EDI-consent	Tradacoms messages
application/EDIFACT	EDIFACT messages
application/EDI-X12	X12 messages
application/octet-stream	Binary messages
application/xml	XML messages

- Message ID. The control ID of EDI documents.
- Conversation ID. Useful for ebXML.

In addition, searches can include or exclude receipts and subordinate messages.

After entering search conditions and executing the search with the **Find** button, you can save the search by name and perform it again with the same conditions. To save a search, type a name in the **Search name** field and click **Save**. To execute a saved search, select the name of the search under **Message tracker > My searches**. Searches are saved on a per-user basis. Searches you have saved are not available to users who log on with a different user ID.

To search by date, expand the **Date** area on the left side below the **Trading information** area. The selection **Don't search by date** means date conditions are not used in searches. To search by date, select **Origination** or **Delivered** and select the date or time conditions.

To adjust the column display of the search results, expand the **Columns** area on the left side below the **Date** area and use the columns controls. Any custom column display you set up applies only to you and not to other users.

Once search results are displayed, you can enlarge the results display area by clicking the **Show/Hide** tab to the left of the **Pick a message** heading.

Message tracker search buttons

The following describes the Message tracker buttons related to searches.

New

New clears any pre-existing search conditions and finds all document records in the database.

Save

Save lets you save the conditions for a search you have performed. This lets you perform the search again without having to set up the conditions again.

To save a search, set conditions for a search and click **Find** to run the search. When the search results are displayed, type a name for the query in the search name field at the top left of the main transaction search page and click **Save**. To perform a saved search, select **Message tracker** > **My Searches** and the name of the search.

Remove

Remove deletes the search identified in the search name field from the My Searches list.

Find

Find searches for all records matching the conditions specified on the search page, if any have been specified.

Message tracker resubmit and delete buttons

There are four buttons at the top of the search results page that control resubmitting and deleting messages.

Resubmit selected messages

Resubmit the messages you have selected by clicking the check box preceding each message. See Resubmitting messages on page 415.

Resubmit search results

Resubmit all messages in the current search results. See Resubmitting messages on page 415.

Delete selected messages

Delete the messages you have selected by clicking the check box preceding each message. This deletes the messages from the database and backup directory.

Delete search results

Delete all messages in the current search results. This deletes the messages from the database and backup directory.

Avoiding possible search conflicts

When you click the find button on the search page, the currently defined search criteria are saved in the database. Each time you return to the search page, the last search criteria you used are presented to you again. This is true even if you restart the server. So in the database are the criteria for the last search you ran plus all of the saved-by-name searches.

If two people are logged on as the same user (for example, admin), each user's last saved query might conflict with the other user's.

If it is important for you to use consistent search criteria, we recommend creating a user account for yourself and logging on as that user.

Message details

Once search results are displayed, you can view a summary list of message activity. Click **Details** to open the message details page for the selected payload or receipt. Trading activity information is provided on three tabs.

Document summary

This tab identifies the message type and status (delivered, negative response, failed, in process, waiting for receipt, resubmitted). If a failed message, a reason is given at the bottom of the tab.

Message processing details

This tab graphically shows the processing steps for the message.

Document activity

This tab provides the dates and times for events related to a message’s processing history.

Icons might appear next to some messages on the search results page. These are:

≠ Indicates the routing ID does not agree with the party profile configuration. This icon can appear in the **From ID** and **To ID** columns, if the columns are selected for search results and if conditions warrant displaying the symbol.

When you move the cursor over the icon a message appears: “Routing ID does not correspond to the receiver” or “Routing ID does not correspond to the sender.”

If the icon appears next to **From ID**, the routing ID does not belong to the partner who sent the message. If the icon appears next to **To ID**, the routing ID does not belong to the community that received the message.

Indicates a message was re-routed.

Resubmitting messages

Resubmitting is a way to reprocess a message to correct an error condition or send a message again to a partner or a back-end system. Buttons for resubmitting messages are at the top of the Message tracker search results page. There also is a message resubmit link at the bottom of each message’s details page. You can view results of resubmitting on the document activity tab.

If you resubmit a message that you already have sent and a partner has acknowledged, the partner’s trading engine might reject it as a duplicate unless steps are taken to anticipate the resend.

If you resubmit a message that you received earlier from a partner, the message is sent again to your community inbound integration delivery exchange.

Receipts cannot be resubmitted.

When a message is resubmitted, Message tracker marks original messages as resubmitted and creates and submits a new message with the content of the original message. The original message and the resubmitted new message are linked together when viewing details in Message tracker.

If an original message was retrieved from an integration pickup exchange configured with message meta-data element values, the original meta-data is resubmitted with the message. If the meta-data configuration for the pickup exchange was changed between the original submission and the resubmission, it is the original meta-data that is resubmitted and not the currently configured meta-data.

Message receipts

Message tracker reports receipts from partners confirming message deliveries, provided the community collaboration settings request partners to send receipts (see Collaboration settings on page 367).

Message receipts also go by other names such as acknowledgments and message disposition notifications (MDNs). In Message tracker, the word receipt is used.

When a signed receipt is requested, but an unsigned receipt is received, the unsigned receipt is rejected (ignored). This results in a resend of the original document until a signed receipt is received or the maximum number of resends is reached.

All signed receipts must contain a Received-Content-MIC value. Any signed receipt that does not contain a Received-Content-MIC is rejected.

When a signed receipt is received, its Received-Content-MIC is compared to the MIC of the original document. This is the case whether the original document requested a signed or unsigned receipt.

When a receipt is received for a document that did not request a receipt, the received receipt is rejected.

Received-Content-MIC values in unsigned receipts are ignored. This is because there is no way to guarantee the validity of any information in an unsigned receipt. And since the primary purpose of the Received-Content-MIC value is for non-repudiation, it does not make sense to give any weight to such a value in an unsigned receipt.

Configure payload view

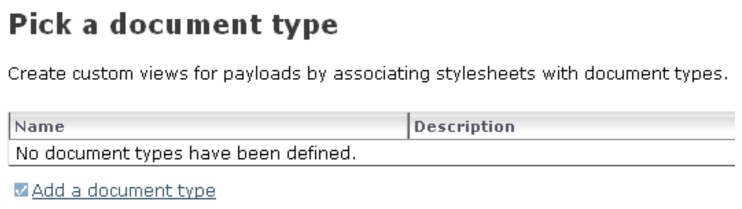
Use this procedure to format EDI or XML documents with stylesheets for viewing in Message tracker. This is optional, except if you process CSOS documents and need to identify CSOS document types. Documents by default can be viewed as plain text without stylesheets.

Steps

1 In Message tracker, expand the **Columns** category in the custom search area on the left side of the page. Select **document type** as a column for display in search results.

2 Execute a search. Note the document type of the document you want to display with a stylesheet. If the document is XML, the document type is blank. See Forcing a document type for XML on page 419 for how to remedy this.

3 Select **Message tracker** > **Configure payload view** on the toolbar. The pick a document type page displays.



Pick a document type

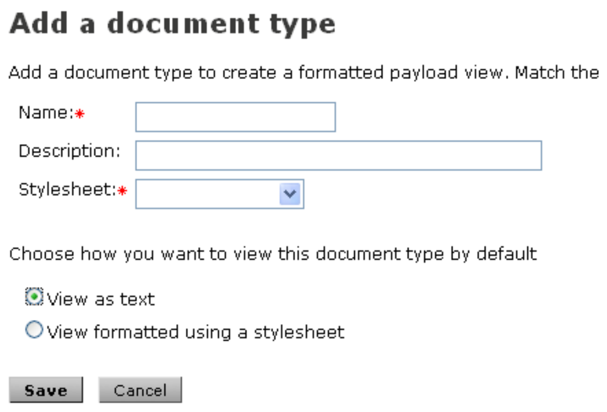
Create custom views for payloads by associating stylesheets with document types.

Name	Description
No document types have been defined.	

Add a document type

Figure 129. Pick a document type page for viewing with stylesheets

4 Click **Add a document type** to display the page of that name.



Add a document type

Add a document type to create a formatted payload view. Match the

Name:*

Description:

Stylesheet:*

Choose how you want to view this document type by default

[✓ View as text]

View formatted using a stylesheet

Figure 130. Add a document type page for viewing with stylesheets

5 In the name field, type the document type you noted in step 2.

6 Optionally, type a description.

7 Select the stylesheet to apply for display from the stylesheet list.

The following stylesheets may be available by default.

defaultEDI.xsl	Formats an EDI document that has been converted to XML via rules files.
defaultXML.xsl	Formats any XML document.
defaultXML2.xsl	Formats any XML document.
e222.xsl	Formats CSOS documents. This stylesheet is intended for CSOS EDI documents.

Depending on your license agreement, other stylesheets may be available. You also can use your own stylesheets rather than the default stylesheets.

For EDI, note that the default stylesheets work only if your system has the proper rules files for converting EDI to XML. These files are in [install directory]\[build number]\conf\tx\oboe_rules. As part of the default installation, you should already have the rules files you need. If not, contact technical support.

For XML documents, if you want to use your own stylesheet, test the stylesheet with a sample of the XML document to make sure it works. Then copy the stylesheet to [install directory]\[build number]\conf\web\documents. Your stylesheet will then appear in the stylesheet selection list in the user interface.

For EDI documents, if you want to use your own stylesheet, contact technical support for help.

8 Select **View formatted using a stylesheet**.

9 Click **Save**.

10 Execute a search in Message tracker. You can type a value in the document type field to narrow the search for a single document type.

11 When the search results are displayed, click a **Details** link for the document type you added to view message details.

12 On the document summary or message processing details tab, click a view payload link. The selected stylesheet is used to display the document. In this view, you have the option to switch to a plain text or browser view or switch back to the stylesheet view.

Forcing a document type for XML

To apply a stylesheet when viewing an XML document, a document type value must be forced in order to identify the documents for applying the stylesheet. In Message tracker, the document type by default is blank for XML documents.

Forcing a document type value will enable you to view future XML documents with a stylesheet, but not the XML documents traded before forcing the value.

For outbound documents

For the integration pickup exchange for XML documents, open the maintenance page for the transport and go to the message attributes tab. Use directory mapping or a fixed attribute to set up DocumentType=XML. (The value does not have to be XML. It is used here for simplicity.)

For inbound documents

For the community delivery exchange for receiving messages from partners, open the maintenance page for the transport and go to the message attributes tab. Set up a fixed attribute value of DocumentType=XML.

❖ ❖ ❖

23 Data backups and deletes

The trading engine copies messages it sends and receives to a backup directory, provided backing up is turned on. Failed messages also are placed in the backup directory; there is not a separate directory for them. (Note that a rejected message is considered a failed message.) Records of traded and failed messages are stored in the database.

The default backup directory is [install directory]\common\data\backup. You can change the directory location.

You can arrange for messages of a certain age to be deleted. This lets you delete unwanted database records and backup files. When triggered, database records about documents and the corresponding files in the backup directory are deleted. Once deleted, you no longer can search for, view or reprocess documents in Message tracker.

When you view payloads in the Message tracker area of the user interface, the system retrieves the documents from the backup directory. When you use Message tracker to resubmit a document, a copy must be present in the backup directory. For these reasons, it is important to ensure the backup directory has the capacity to store all the documents you want.

Delivery exchange configuration controls whether documents are backed up. See Transport maintenance on page 247 for more information.

Concepts

- Change backup directory
- Delete unwanted files and records on page 423
- Purge all records, backup files on page 424

Change backup directory

The backup directory stores copies of all traded documents in their plain text and encrypted forms.

Go to the trading configuration area of the user interface and click **Set the backup directory path** to change the backup directory.

Root path

Use the **Root path** field to specify the path of the backup directory. The default is [install directory]\common\data\backup.

The system supports conventional paths for the backup directory, such as C:\data\backup in Windows, and paths incorporating a server name (Uniform Naming Convention).

Schedule to create backup subdirectories

The **Schedule to create backup subdirectories** field specifies the schedule for creating backup subdirectories. The trading engine creates a subdirectory periodically to limit the number of backup files in a single directory. **Automatic**, the default setting, causes a backup subdirectory to be created about every hour. Subdirectories are created only as needed to accommodate trading activity. Subdirectories are not added when there is no activity.

The automatic setting is fine in most cases. However, if your volume is low (10,000 documents a month or less) and you want to minimize the number of subdirectories created, specify the infrequent value **Monthly**. If your volume is extremely high (500 documents per minute or more) specify the high frequency **Every minute** to reduce the number of files in each subdirectory and improve performance.

Although the setting **Never** is available, it is not recommended.

Because you use Message tracker to search for, view and resubmit documents, there is no need to manually examine the backup directory contents. If you open the directory, however, you see the system appends names of backed up files with three labels. These are:

Consumed

Consumed is appended to names of documents that have been received from partners but not unpackaged or that the trading engine has acquired but not packaged or sent.

UnpackagedRequest

UnpackagedRequest is appended to names of documents that have been received from partners and unpackaged. These documents are plain text.

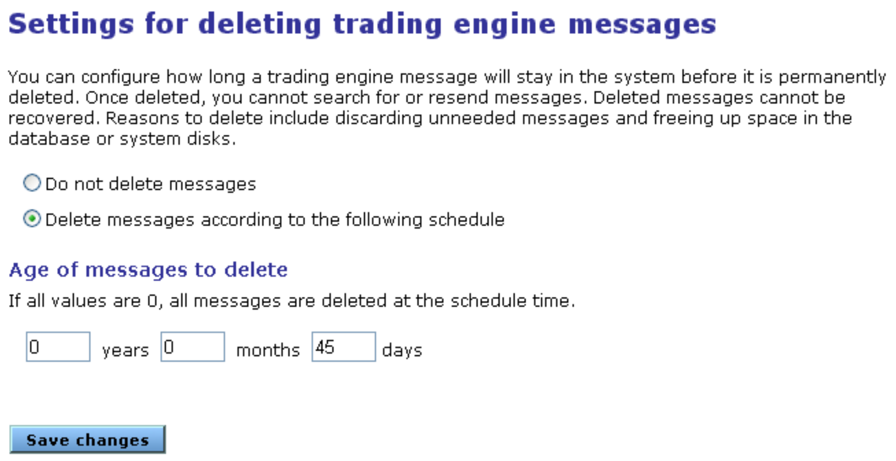
Packaged

Packaged is appended to names of documents that have been packaged and scheduled to be sent to partners.

Delete unwanted files and records

You can configure the trading engine to delete unwanted database records and files in the backup directory based on the age of the records and files.

Go to the trading configuration area of the user interface and click **Configure settings to delete trading engine messages**. This displays the settings for deleting trading engine messages page (Figure 131).



Settings for deleting trading engine messages

You can configure how long a trading engine message will stay in the system before it is permanently deleted. Once deleted, you cannot search for or resend messages. Deleted messages cannot be recovered. Reasons to delete include discarding unneeded messages and freeing up space in the database or system disks.

○ Do not delete messages

● Delete messages according to the following schedule

Age of messages to delete

If all values are 0, all messages are deleted at the schedule time.

0 years 0 months 45 days

Figure 131. Settings for deleting trading engine messages page

The default configuration is to delete database records and backup files after they have been in the system for 45 days. You can use the years, months and days fields to change the age settings. The system checks every 15 minutes for records and documents of this age to delete.

You can choose never to delete any database records or backup files. This is not recommended unless you have requirements for maintaining database records and backup files indefinitely.

An event such as the following is written to the event log file when a message is deleted:

```
2005/10/31 08:06:08.812 High
Administration.Persistence.MessagesPurgedMessage
Deleted(CoreId(1130771141162.903@HOSTNAME))
```

Purge all records, backup files

You can delete all database records and files in the backup directory at once using a utility named **messagePurgeTool**. This command-line utility is in [install directory]\[build number]\tools, and must be run from that directory.

CAUTION: Make sure the trading engine server is turned off before using messagePurgeTool.

To delete all records and backup files, run the following command:

```
messagePurgeTool -deleteAll
```

This command deletes all database records and backup files regardless of age or state. Depending on the volume of records and files, the utility may have to run a while to delete all.

The utility also has a **resetMessages** parameter. This is for updating purge settings for records and backup files if you change the age settings on the settings for deleting trading engine messages page. See Delete unwanted files and records on page 423. The utility may take a while to run if there are a large number of records to reset.

Details of running messagePurgeTool are written to the messagePurgeTool.log file in the logs directory.

❖ ❖ ❖

24 FTP client scripting interface

The FTP client in the trading engine includes a scripting interface to accommodate non-standard interaction with FTP servers. The default FTP client implementation uses an XML file, called the command set document. The XML file defines meta-commands comprised of one or more FTP commands to be sent to an FTP server. The default command set document works with most FTP servers without any modifications.

The following topics describe how to change the default FTP client implementation when the client must interact with an FTP server that requires non-standard commands or expects commands in a different order than the default in the command set document. Such changes can range from editing the command set document to creating Java classes that implement non-standard FTP commands.

Any change to the FTP client requires familiarity with the trading engine and the FTP protocol as described in RFC959. Simple modifications require editing the XML command set document. Advanced modifications require familiarity with the Java programming language and writing and compiling Java classes.

Concepts

References

- RFC959
  http://www.ietf.org/rfc/rfc959.txt

- Draft proposal for stream mode restarts
  http://www.ietf.org/internet-drafts/draft-ietf-ftpext-mlst-16.txt

Levels of scripting

The following describe the three levels of possible scripting changes, ordered from simplest to most complex.

1 **Edit the command set document.** At this level, you edit the default command set document to change the order of commands sent to the FTP server or remove commands. For example, if an FTP server immediately prompts for a password rather than first prompting for a user name, you can remove the line that sends the User command from the authenticate meta-command.

2 **Change Java classes containing commands.** A Java class implements each command (for example, User). To change the behavior of a command or to add a command, you must edit or create the Java class that implements the command. Then compile the class and make it available to the FTP client.

3 **Write a custom FTP client implementation.** The default client implementation is the framework of the FTP client. It includes classes that manage the connections with the server and that read and execute the commands in the script. You can replace the default client implementation with one that does not use a script. For example, if a user has an FTP client implementation written in Java, it could be modified to work with the trading engine by replacing the default client implementation with an interface to the user's existing implementation.

The following topics describe each level of scripting in greater detail.

Editing the command set document

The script that specifies commands sent to the host is the XML command set document, named ftpcommandset.xml in [install directory]\[build number]\conf. It contains a set of meta-commands, each consisting of one or more FTP commands to be sent to an FTP server.

While interacting with the FTP client, the trading engine executes meta-commands defined in the script, such as receive and send. For each meta-command, the script specifies the FTP commands to send to the FTP server and in what order. The trading engine is not aware of the specific FTP commands being sent to the FTP server, which means changes to the command set document do not require changes to the trading engine.

One example of the changes you can make to the command set document is reordering the FTP commands comprising a meta-command. As a simple example, the receive meta-command sends the Type and Size commands, in that order. You could reverse the order by transposing those two lines in the script.

Users of the FTP client, such as the ftpTester program and the trading engine itself, invoke the connect and authenticate meta-commands before issuing any other commands.

Changes made to the script take effect the next time the trading engine needs to access the FTP server; the trading engine server does not have to be restarted.

To use a command set document other than ftpcommandset.xml, you must add an entry for your document in [install directory]\[build number]\conf\filereg.xml and restart the trading engine server. You also must do this if you want to use a different command set document for each FTP delivery exchange.

Changing Java classes

When re-arranging commands in the command set document is insufficient for your needs, you might need to edit a command or create a command and add it to the command set document. Changes of this type require knowledge of the Java programming language and an understanding of the default implementation of the commands. For example, the delete meta-command is implemented by a single FTP command, Dele. The default implementation of this command is available from technical support.

Presuming you have obtained the source code for the default command implementations, the Java source file indicates that Dele extends FtpCommand. Similarly, the Java source file for FtpCommand indicates that it extends AbstractFtpCommand. Figure 132 shows the class hierarchy.

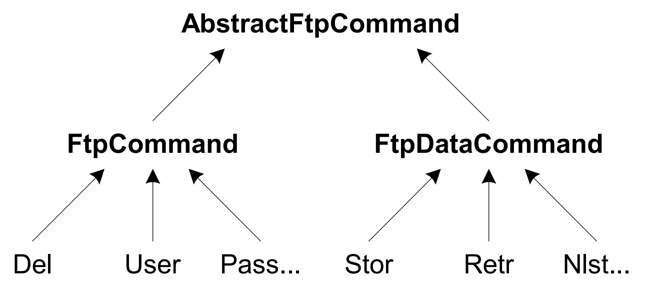


Figure 132. Class hierarchy

FtpCommand and FtpDataCommand are separate due to the FTP protocol requirement for a separate control connection and data connection.

Commands that can potentially return large volumes of data, such as Stor, Retr and Nlist, extend FtpDataCommand. Other commands extend FtpCommand, and they use only the control connection. Dele falls into this category. RFC959 defines which commands do and do not use the data connection.

A review of Dele.java reveals the Dele class implements the getCommand method, which returns the command string to be sent to the server. All command classes must implement getCommand because it is abstract in the base class.

In addition, Dele overrides the validateParameters method in AbstractFtpCommand. This allows Dele to ensure its argument is not empty. Many other commands that require an argument also override validateParameters for the same reason.

The base classes contain other methods, such as execute, that can be overridden. The details of how and when to do this should become evident after examining the contents of the base classes and the various command classes that extend them.

Writing a custom FTP client

If your requirements cannot be met using the methods already described, you can implement your own FTP client. The client you write must extend the FtpClient base class (which is abstract), allowing the trading engine to interact with your client. The methods in your implementation must accept input parameters and return output values in the same form as the default implementation. To ensure conformity, use the source code in the default implementation as a reference. The source code is available from technical support. You might want to hard-code the commands sent to the server instead of using a command set document, as does the default implementation. This will make your implementation simpler.

If you already have an FTP client implementation written in Java that you would like to use with the trading engine, it must extend the abstract FTP client base class.

After creating and compiling an FTP client implementation, tell the trading engine to use it. This is done by defining a property called com.cyclonecommerce.ftp.client on the JVM invocation line in the startServer script in the bin directory (startServer.ini if running as a Windows service). This property must contain the name of the class that implements FtpClient. For example:

```
com.cyclonecommerce.ftp.client=com.me.MyFtpClient
```

Make sure the directory containing your class is in the classpath specified in the environment file in the bin directory. For example, you could put the class in:

```
[install directory]\[build number]\classes\com\me\MyFtpClient.class
```

For detailed information about how the trading engine finds the FTP client implementation, see FtpClientFactory.java, which is available from technical support.

All FTP delivery exchanges must use the same FTP client implementation because it is a system property in the JVM.

FTP tester tool

You can use the ftpTester tool to exercise the FTP client outside of the trading engine. The script to start ftpTester can be found in [install directory]\[build number]\tools.

ftpTester is a console-based application that can verify the operation of the FTP client in the trading engine and a partner's FTP server. The trading engine server does not have to be running to use this tool. You can use it on UNIX or Windows.

ftpTester duplicates the way the trading engine accesses an FTP server. It is a test program to verify interoperability with FTP servers. If you can send, list, receive and delete files on a FTP server using ftpTester, this is a good indication the trading engine can interoperate with the server.

ftpTester prompts for all the information it needs, as the following illustrates. Two views are shown, depending on whether you are testing receiving (consuming) or producing (sending).

Consumer options

```
**** Welcome to the Cyclone FTP test program ****
-> Enter host: localhost
-> Enter user: ftpuser
-> Enter password: cyclone
-> Enter account (leave blank for most servers):
-> Enter pickup directory (blank for "pickup"):
-> Enter c for CONSUMER client (list, receive, delete), p for PRODUCER
(send). (Blank assumes c): c
-> Should temp files be used to avoid read/write collisions? (Y/N -
default is Y):
-> Should a separate inbox dir be used (instead of putting temp files
in the pickup dir)? (Y/N - default is Y):
-> Enter inbox dir where files will initially be uploaded before being
moved to pickup dir (blank for "inbox"):
-> Use SSL? (Y/N - default is N):
```

```
-> Should restarts be attempted for binary files if supported by the
host? (Y/N - default is Y):
-> Enter minimum bytes a file must contain to be eligible for restarts
(blank for 100000):
FtpClientSettings - host=localhost pickupDir=pickup ctlPort=21
passive=true type=Binary mode=Stream struct=File user=ftpuser account=
transportId=FtpTester receiveTempDir=/Applications/cyclone/b1095/bin/
ftpRestart_FtpTester connectTimeoutMs=30000 readTimeoutMs=30000
attemptRestarts=true evaluateRestartable=true
restartableMinBytes=100000 commandSet=../conf/ftpcommandset.xml
preserveFilename=true overwriteIfDupe=true fixOutputFilenames=false
tempFileExt=.tmp useDebounce=true useInbox=true isSSLEnabled=false
Host working directory: "/Users/ftpuser" is the current directory.
Host pickup directory (used by REC, DEL and LIST): pickup
Local working directory (used by REC and LLIST): /Applications/
cyclone/b1095/bin (LCD to change)

-> Enter CONSUMER command (e.g. ?, LIST, RECeive, DELete, LLIST, ASCII,
BIN, LCD, QUIT): ?
CONSUMER metacommands (abbreviations shown in upper case):
?                       help
LIST                 list files on host
RECeive filename  retrieve file from host
DELete filename   delete file from host
ASCII             perform ASCII data transfers
BINary            perform binary data transfers
LCD               change local working directory
LLIST             list files in local working directory
PASV              perform data transfers in passive mode (the default)
PORT              perform data transfers in port mode
QUIT/EXIT/BYE     exitNormal FtpTester
Note:  Metacommands are scriptable via conf/ftpcommandset.xml
Host working directory: "/Users/ftpuser" is the current directory.
Host pickup directory (used by REC, DEL and LIST): pickup
Local working directory (used by REC and LLIST): /Applications/
cyclone/b1095/bin (LCD to change)

-> Enter CONSUMER command (e.g. ?, LIST, RECeive, DELete, LLIST, ASCII,
BIN, LCD, QUIT):
```

Producer options

```
**** Welcome to the Cyclone FTP test program ****
-> Enter host: localhost
-> Enter user: ftpuser
-> Enter password: cyclone
-> Enter account (leave blank for most servers):
-> Enter pickup directory (blank for "pickup"):
-> Enter c for CONSUMER client (list, receive, delete), p for PRODUCER
(send). (Blank assumes c): p
-> Should original filenames be preserved when sending? (Y/N - default
is Y):
-> Should existing files be overwritten? (Y/N - default is Y):
-> Should temp files be used to avoid read/write collisions? (Y/N -
default is Y):
-> Should a separate inbox dir be used (instead of putting temp files
in the pickup dir)? (Y/N - default is Y):
-> Enter inbox dir where files will initially be uploaded before being
moved to pickup dir (blank for "inbox"):
-> Use SSL? (Y/N - default is N):
-> Should restarts be attempted for binary files if supported by the
host? (Y/N - default is Y):
```

```
-> Enter minimum bytes a file must contain to be eligible for restarts
(blank for 100000):
FtpClientSettings - host=localhost pickupDir=pickup ctlPort=21
passive=true type=Binary mode=Stream struct=File user=ftpuser account=
transportId=FtpTester receiveTempDir=/Applications/cyclone/b1095/bin/
ftpRestart_FtpTester connectTimeoutMs=30000 readTimeoutMs=30000
attemptRestarts=true evaluateRestartable=true
restartableMinBytes=100000 commandSet=../conf/ftpcommandset.xml
preserveFilename=true overwriteIfDupe=true fixOutputFilenames=false
tempFileExt=.tmp useDebounce=true useInbox=true isSSLEnabled=false
Host working directory: "/Users/ftpuser" is the current directory.
Host inbox directory (used by SEND for uploading, after which files are
moved to pickup): inbox
Host pickup directory (used by SEND and LIST): pickup
Local working directory (used by SEND and LLIST): /Applications/
cyclone/b1095/bin (LCD to change)

-> Enter PRODUCER command (e.g. ?, SEND, LLIST, ASCII, BIN, LCD, QUIT):
?
PRODUCER metacommands (abbreviations shown in upper case):
?                  help
SEND filename      write file to host
ASCII              perform ASCII data transfers
BINary             perform binary data transfers
LCD                change local working directory
LLIST              list files in local working directory
PASV               perform data transfers in passive mode (the default)
PORT               perform data transfers in port mode
QUIT/EXIT/BYE      exitNormal FtpTester
Note:  Metacommands are scriptable via conf/ftpcommandset.xml
Host working directory: "/Users/ftpuser" is the current directory.
Host inbox directory (used by SEND for uploading, after which files are
moved to pickup): inbox
Host pickup directory (used by SEND and LIST): pickup
Local working directory (used by SEND and LLIST): /Applications/
cyclone/b1095/bin (LCD to change)

-> Enter PRODUCER command (e.g. ?, SEND, LLIST, ASCII, BIN, LCD, QUIT):
```

After prompting for the initial configuration information such as the host, user and password, FtpTester displays a main prompt that allows you to enter meta-commands to perform simple operations such as list, send and receive. You can enter a question mark (?) at this point to get more information.

❖ ❖ ❖

25 Test trading

After setting up a community profile, you are ready to add trading partners and begin trading. While this is appropriate for some users, others may want to test their system first. You can perform test trading with a trading partner.

Procedure

- Configure for test trading
- Start the test on page 435
- Troubleshooting for e-mail testing on page 436
- Complete the testing on page 436

Configure for test trading

Use this procedure to configure the trading engine for test trading.

We recommend creating a temporary community profile and a self-signed certificate for the tests rather than use the profile and certificate that you would use for actual production trading of documents. You can retire the temporary profile and certificate after the tests.

Steps

1. Complete your test community profile. See Add a community on page 146.

   For information about transport options see Delivery exchanges on page 181.

   You and your partner should decide which transport method to use. Then complete your community profile to receive messages via the appropriate transport method. For simplicity, we suggest both of you first use the same transport method to receive and send messages.

   For simplicity, unless you have other integration plans, have the community use file system integration to pick up outbound messages and send inbound messages to the back end. See File system transport on page 200.

2. Generate a test self-signed certificate for your test community profile. See Set up certificates for a community on page 335.

3 Export your community profile to a file. You can send this file to your testing partner as an attachment to an e-mail. If you send your profile by FTP, make sure to use a binary transfer.

If you send the profile file as an e-mail attachment, we recommend using a utility such as WinZip to compress the file before sending. This is to protect the file from possible corruption during transmission. Some SMTP servers append verbiage to e-mail attachments, which can harm the profile file.

4 Obtain your partner's profile.

Have the partner send the profile file to you by e-mail or another method. Then import the profile. If the partner also uses Cyclone Activator, the profile file should include the partner's public key and certificate.

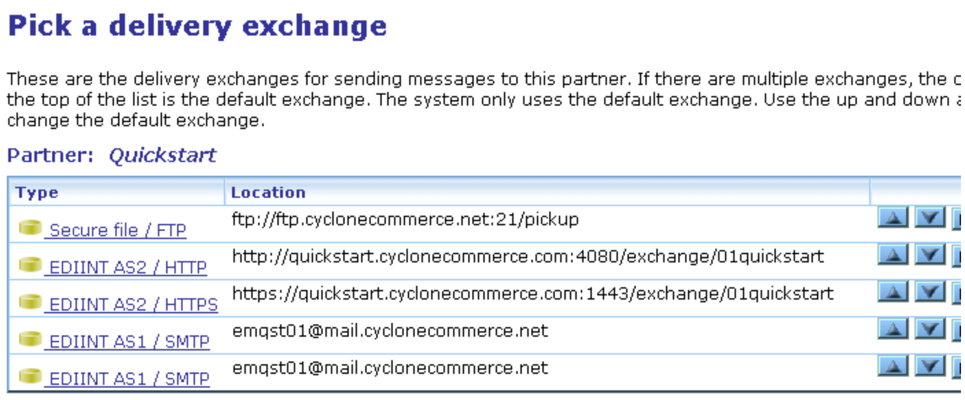
If the partner uses other interoperable software, create the profile in the trading engine using information supplied by your partner. The partner also must send the certificate and public key in a file.

For more information see:

Add a partner on page 150
Import a partner profile on page 152.
Import certificates for partners on page 352

5 In the partner profile, select a default transport for sending test documents. If there is only one transport, that is the default.

To check available transports in the partner profile, open the partner summary page and click **Delivery exchange** on the navigation graphic to display a list of available transports for sending. If there is more than one to choose from, you and your partner should decide which one to use. Use the up and down arrow keys to move the desired transport to the top of the list. The transport at the top of the list is the default method.



Pick a delivery exchange

These are the delivery exchanges for sending messages to this partner. If there are multiple exchanges, the c
the top of the list is the default exchange. The system only uses the default exchange. Use the up and down a
change the default exchange.

Partner: *Quickstart*

Type	Location
Secure file / FTP	ftp://ftp.cyclonecommerce.net:21/pickup
EDIINT AS2 / HTTP	http://quickstart.cyclonecommerce.com:4080/exchange/01quickstart
EDIINT AS2 / HTTPS	https://quickstart.cyclonecommerce.com:1443/exchange/01quickstart
EDIINT AS1 / SMTP	emqst01@mail.cyclonecommerce.net
EDIINT AS1 / SMTP	emqst01@mail.cyclonecommerce.net

Figure 133. Example of available transports for sending in partner profile

6 Contact your trading partner to confirm test arrangements.

Start the test

Before you begin testing you need some test files on hand. For efficient testing, each of these documents should be initially 10 KB or less in size. Once you have traded a number of documents, you can increase the size of the test documents.

You can use your own documents or you can use the Document Generator utility to generate EDI or XML documents of any size and at any rate. See Document Generator on page 437 for how to create test documents.

Steps

1 Start the Server application and log on.

2 To follow trading activity, go to the Message tracker area of the user interface. For information about this feature see Message tracker on page 411.

3 If you are using file system integration, place one test EDI document in the community outbound integration directory. If you are using another integration method, the trading engine must be able to retrieve the document through that delivery exchange.

The trading engine processes these documents and sends them to your trading partner. Look at Message tracker for notification that a message has been sent. Upon receiving the message, the partner should send a receipt to your community.

If you are using file system integration, the trading engine writes the inbound document to the community integration delivery exchange.

4 After sending and receiving some EDI documents, try trading some XML or binary documents.

Troubleshooting for e-mail testing

If you are using e-mail to send or receive messages and the transport does not perform correctly during testing, try to determine the cause of the difficulty by checking the following items:

- Determine whether it is a sending or receiving problem.

  If a sending problem, is the trading engine correctly connected to the SMTP server?

  If a receiving problem, is the trading engine correctly connected to the POP server?

- Check your e-mail account by sending a message from another e-mail account to the community e-mail account for receiving messages.
- Is your connection to the Internet functioning correctly?
- Determine whether the problem is related to encryption or decryption. If encryption related, can you verify whether the mail server is S/MIME compliant?
- Determine whether the problem is related to a trading partner's profile settings.
- Does your organization's network infrastructure allow e-mail attachments?

Complete the testing

After you have completed your testing, document the test. Include details about:

- Any changes you made to your system.
- Any customization of the trading engine that you did and how to recreate it, if necessary.
- If you encountered problems, any details that might help someone diagnose and correct similar problems in the future.

❖ ❖ ❖

26 Document Generator

The Document Generator utility is included with the trading engine. You can use it to create test documents that conform to the structures of X12 EDI or BizTalk XML formats. To create an end-to-end test, you can generate documents of any size and send them at any interval you choose to another trading engine.

Procedure

- Create EDI or XML test documents on page 437
- Run from a command line on page 440

Create EDI or XML test documents

Use this procedure to create EDI or XML test documents in Document Generator and put them in an output directory.

You can run multiple sessions of the Document Generator. Each session can generate different document types, sizes and rates.

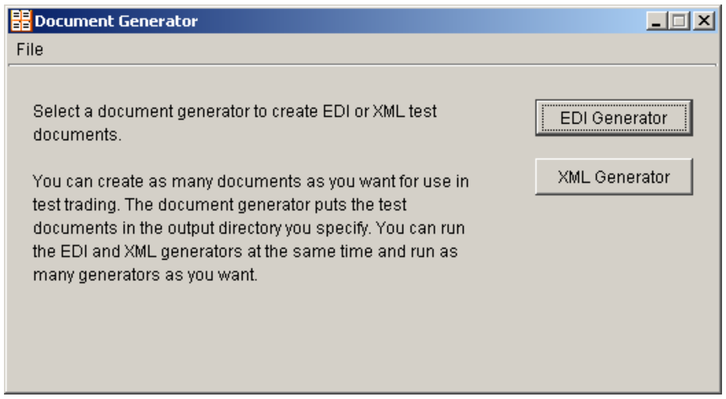
Steps

1. In Windows select **Start** > **Programs** > **Cyclone** > **Document Generator**.

   In UNIX log in to the cyclone account you created previously. Ensure that you have X Windows connectivity to the server where you installed the application. Run the following command to open the Document Generator:

   [install directory]/[build number]/bin/docGen

   You also can run the Document Generator from a command line. See Run from a command line on page 440.



Select a document generator to create EDI or XML test documents.

You can create as many documents as you want for use in test trading. The document generator puts the test documents in the output directory you specify. You can run the EDI and XML generators at the same time and run as many generators as you want.

Figure 134. Document Generator window

2 Click **EDI Generator** or **XML Generator** to open the EDI or XML Document Generator window. The two windows are similar, but only the EDI window has Control ID and Input template fields.

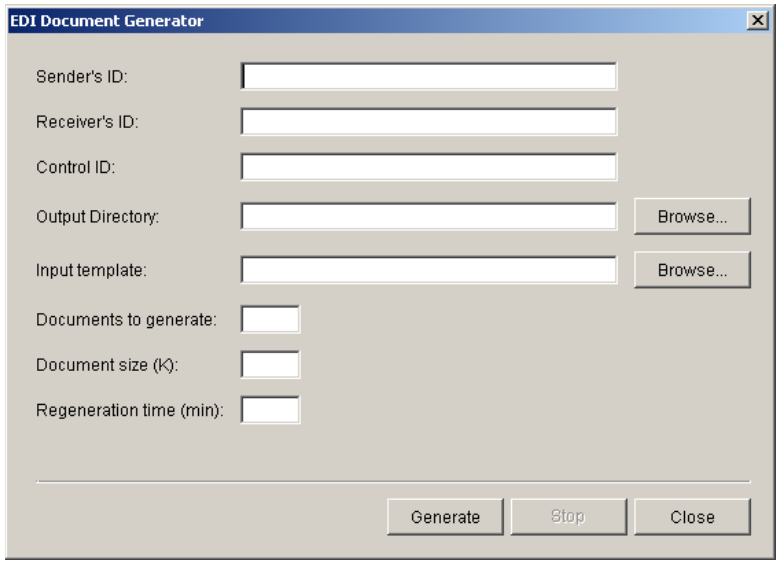


Figure 135. EDI Document Generator window

3 Complete the fields. See Field descriptions on page 438.

4 Click **Generate** to generate the number and size of documents you specified. The Document Generator continues to generate documents at the interval you specified until you click **Stop** or close the EDI or XML Document Generator window.

Field descriptions

The following describes the fields on the EDI and XML Document Generator windows. For procedure see Steps on page 437.

Sender’s ID

Type the ID of the sender.

Receiver’s ID

Type the ID of the receiver.

Control ID (EDI only)

Type any numeric control ID. This is the starting number for the document counter.

Output Directory

Type the directory where the Document Generator writes the outbound documents. Or, use the **Browse** button to locate this directory. This is typically the sender’s EDI or XML out directory.

Input template (EDI only)

If you want to use your own X12 EDI document as the template for creating test EDI documents, click **Browse** to point to the document on your system. Document Generator copies your document and inserts your specified sender, receiver and control ID in the generated test documents.

If you want Document Generator to create documents for you, leave this field blank.

Documents to generate

Type any value between **1** and **999999** to indicate the number of documents you want to create per unit of time. The Document Generator creates all of these documents at once.

Document size (K)

Type any value between **1** and **999999** to indicate the size of each document you want to create.

Regeneration time (min)

Type any value between **1** and **999999** to indicate the time in minutes the Document Generator waits to create the next document or set of documents.

Run from a command line

You can use Document Generator from a command line without the graphical user interface (GUI). You do this by running a command with parameters for the test documents you want to create. The command is **docGen**.

You cannot pause the Document Generator from the command line as you can when using the GUI. Only one Document Generator at a time can be started from the command line in a single DOS window or terminal window.

Note: If you run the trading engine on UNIX and there are spaces in the sender's or receiver's ID, we recommend using the Document Generator GUI. See Create EDI or XML test documents on page 437.

Command line parameters

The following table shows the command line parameters for the Document Generator. You do not have to use the parameters in the order listed.

Parameter	Description	Usage
-type	Valid document types are **EDI** or **XML**.	Required
-sender	ID of the sender.	Required
-receiver	ID of the receiver.	Required
-docid	The number to use as the control ID of the first EDI document to be generated. Do not use for XML documents because they do not have control IDs.	Required for EDI N/A for XML
-outpath	The directory where the Document Generator writes the outbound documents. This is typically the sender's EDI-out or XML-out directory.	Required
-size	Any value between **1** and **999999** to indicate the size of each document you want to create.	Required

Parameter	Description	Usage
-ndocs	Any value between **1** and **999999** to indicate the number of documents you want to create per unit of time. The Document Generator creates all of these documents at once.	Required
-infile	The path to the EDI document on your system that you want to use as the template for generating test documents. You can use copies of your own EDI document as the test documents rather than the test documents that Document Generator creates for you. If you use your own EDI document as the template, Document Generator copies it and inserts your specified sender, receiver and control ID.	Optional for EDI N/A for XML
-interval	Any value between **1** and **999999** to indicate the time in minutes the Document Generator waits to create the next document or set of documents. If you do not use a value, Document Generator creates the number of documents specified by -ndocs once. If you use a value, Document Generator creates the specified number of documents at the specified interval until you stop the tool.	Optional
-h, -help or ?	Displays a list of the parameters. (In Windows, see docgen.log in logs directory for list.)	Optional

Command line format

The following are examples for running Document Generator from a command line. Be sure you run the utility from the trading engine bin directory.

UNIX

For UNIX, the following example shows the command line format for Company1 to create 7 EDI documents that are 3K in size every 5 minutes and place them in the EDI out directory for sending to Partner1. The control ID is 302.

```
./docGen -type edi -sender company1 -receiver partner1 -
docid 302 -outpath /home/cyclone/ci400/data/company1/ediout
-size 3 -ndocs 7 -interval 5
```

If you run the docGen command without any parameters, the GUI opens.

To stop the generator:

1. Run **ps -ef | grep java** to find the process ID (PID).
2. Run **kill -9 [PID]**, where PID is the number found in the previous step.

Windows

For Windows, the following example shows the proper command line format. Events related to running the tool are written to the docgen.log in the application's log directory.

```
docGen -type edi -sender company1 -receiver partner1
-docid 302 -outpath [installation_directory]/data/company1/
ediout -size 3 -ndocs 7 -interval 5
```

Press **Ctrl-C** in the DOS window to stop the Document Generator.

If there are spaces in the sender's or receiver's ID or out directory name, place the IDs or directory name in quotation marks so Windows properly handles the spaces.

❖ ❖ ❖

27 ebXML support

ebXML, sponsored by UN/CEFACT and OASIS, is a modular suite of specifications that enables a company located anywhere to conduct business over the Internet. ebXML embodies the definition and registration of processes for exchanging business messages, conducting trading relationships and communicating data in common terms.

The trading engine supports the Collaboration-Protocol Profile and Agreement Specification 2.0, available at http://www.oasis-open.org/committees/ebxml-cppa/documents/ebcpp-2.0.pdf (Adobe Acrobat required). The supported CPA schema is http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd.

Your organization must have a thorough understanding and working knowledge of ebXML to successfully trade documents using this message protocol. For information about ebXML see www.ebxml.org or www.oasis-open.org. Books about ebXML also are available commercially.

The CPA represents a contract between two parties defining how they will trade ebXML messages. The trading engine takes directions from the CPA, such as the transport to use for sending messages and whether outbound messages are signed and encrypted. The trading engine also enforces the signing and encryption requirements in the CPA for inbound messages.

Concepts

Procedure

ebXML message lifecycle

The following topics summarize inbound and outbound processing of ebXML messages by the trading engine.

Outbound ebXML processing

The following summarizes the trading engine processing steps for an outbound ebXML message (Figure 136). There are variations to this description, but the following is a typical case.

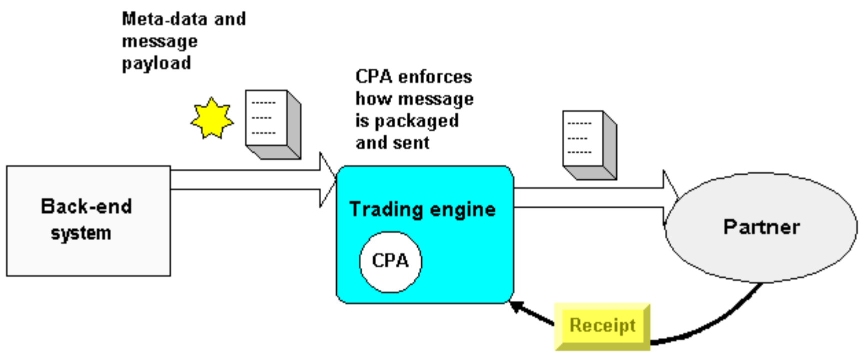

Meta-data and message payload
CPA enforces how message is packaged and sent
Back-end system
Trading engine
CPA
Partner
Receipt

Figure 136. Simplified view of ebXML outbound processing

1 The back-end system produces meta-data and a payload. The meta-data identify the CPA to use for packaging the outbound message.

2 The trading engine receives or acquires the message. The way this is accomplished depends on the integration method. For instance, if file system integration is used, the trading engine polls the back-end file system. If a message meta-data document (MMD) is present, it retrieves the MMD. The MMD tells the trading engine where to poll on the file system to retrieve the payload. MMDs, which are meta-data carriers, are explained in Message meta-data documents on page 452.

Multiple payloads might accompany the same meta-data. If so, the trading engine keeps track of all payloads and relates them to the same meta-data. (Multiple payloads are not supported when JMS is the integration method for outbound messages.)

3 The trading engine validates the meta-data against the CPA the meta-data identifies.

4 The trading engine collaboration manager uses the CPA the meta-data identifies to build the binary collaboration rules for packaging and sending the ebXML message to a partner. Packaging usually involves encrypting and signing the message. The collaboration settings in the trading engine user interface are not used.

5 The trading engine sends the message. This is a SOAP message, consisting of a SOAP header containing collaboration meta-data and a payload attachment. If multiple payloads were received from integration, a single SOAP message is produced, with each payload as an attachment in the message.

6 When reliable messaging is engaged in the CPA, the trading engine receives a receipt acknowledging that the partner received the message. The trading engine checks the receipt against the CPA to see whether an acknowledgment was requested and whether the receipt conforms to the CPA (for example, the CPA might require signed receipts). The trading engine matches the receipt with the outbound message and marks the message as delivered.

If the outbound message had multiple payloads, the trading engine associates the same receipt to all payloads, which appear as separate records in Message tracker.

The receipt is a message handler signal and is not produced to the back-end system.

Inbound ebXML processing

The following summarizes the trading engine processing steps for an inbound ebXML message (Figure 137). There are variations to this description, but the following is a typical case.

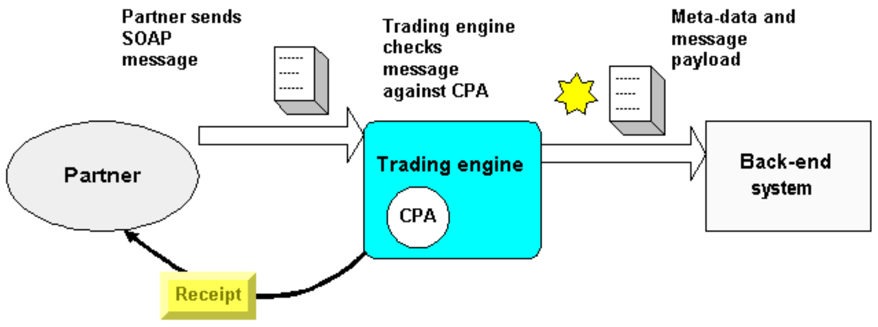

Partner sends SOAP message
Trading engine checks message against CPA
Meta-data and message payload
Partner
Trading engine
CPA
Back-end system
Receipt

Figure 137. Simplified view of ebXML inbound processing

1. The trading engine receives a SOAP message from a partner. The message has collaboration meta-data and one or more payload attachments. The meta-data identify the CPA to use for this payload.

2. The trading engine validates the message against the CPA identified in the inbound message. For example, if the CPA requires, the trading engine determines whether the message is a duplicate to one received earlier. If the message is a duplicate, the trading engine re-sends the earlier transmitted receipt to the partner, but does not produce the message again to the back-end system.

   The trading engine also determines how many payloads are in the message. If there are multiple payloads, the trading engine keeps track of all payloads and relates them to the same meta-data.

3. If the CPA requires reliable messaging, the trading engine sends the partner a receipt that acknowledges the message has been received. If the inbound message had multiple payloads, the trading engine associates the same receipt to all payloads, which appear as separate records in Message tracker.

4. The trading engine sends the payload to a back-end system via an integration transport. Depending on the transport, the trading engine might produce meta-data for the payloads. See Supported integration transports on page 456.

ebXML message meta-data

To comply with ebXML standards, the trading engine supports message meta-data elements with message payloads. The meta-data elements are information about message payloads, such as the identity of the CPA to use for message processing.

Meta-data are exchanged between the trading engine and a back-end system via the following integration transports:

- JMS (inbound and outbound integration)
- File system (outbound integration)
- File system with message meta-data (inbound integration)
- Web services API server (outbound integration)
- Web services API client (inbound integration)

Supported integration transports on page 456 explains the role of these transports in more detail.

ebXML meta-data descriptions

The following describes the meta-data elements.

These elements are listed in the correct format. When using meta-data elements, make sure to use the proper case.

AckRequested

Indicates whether a partner is asked to send a receipt upon receiving the message. Valid values are **true** and **false**. This element can be set per message.

AckSignatureRequested

If AckRequested is true, this indicates whether the partner is asked to sign the receipt. Valid values are **true** and **false**. This element can be set per message.

Action

Identifies an action within a Service that processes the message. The Action must be unique within the Service in which it is defined. The Service designer defines the value of the Action element.

ConversationId

A string identifying all consecutive messages belonging to the same collaborative business process instance. This is useful for monitoring and auditing purposes.

a set of related messages that comprise a conversation between two parties. It must be unique in the context of the specified CPA ID.

CPAId

Identifies the CPA that governs the collaboration between the trading parties. This matches the CPA ID in the CPA, not the name of the CPA XML file.

DuplicateElimination

Indicates whether the receiving message service handler should accept or reject duplicate messages. Valid values are **true** and **false**. This element can be set per message.

ebXML.Ack

A message service level acknowledgment message.

ebXML.AsyncMshSignal

Indicates that the ebXML message service level signal is used asynchronously.

ebXML.DigestAlgorithm

Defines the algorithm for computing the digest of the message.

ebXML.EncryptionRequested

Indicates whether the message must be encrypted.

ebXML.Error

Used for an ebXML message service level error message.

ebXML.Fault

Used for an ebXML message service level fault message.

ebXML.IsSyncResponse

Indicates a synchronous response.

ebXML.MshSignalType

Indicates the type of the message service level message.

ebXML.Ping

Used for a message service level ping message. On success, a pong message service level message must follow.

ebXML.Pong

Used for a message service level pong message. A pong message must be sent after having received a ping message service level message.

ebXML.ReceiveAction

Indicates the action taken.

ebXML.SignalType

Sets the message service level action of the message. The action can be an acknowledgment, error, message status request.

ebXML.SigningRequested

Indicates whether the ebXML message must be digitally signed.

ebXML.StatusRequest

Used to make a request about a previous ebXML message.

ebXML.StatusResponse

The response to a request about a previously sent ebXML message.

ebXML.StatusResponse.RefToMsgId

References the message the requesting message service level message was inquiring for status.

ebXML.SyncMshSignal

Indicates the message service level signal is used synchronously.

FromRole

The role of the sending party.

Hl7Version

Indicate the version of the HL7 message.

IsHl7

Indicates the payload is an HL7 message.

IsStarBOD

Set to **true** for business object documents (BODs) that conform to Standards for Technology in Automotive Retail (STAR). Otherwise, set to **false**. See STAR BODs with ebXML on page 480.

MessageId

A unique identifier for a message that conforms to RFC 2822.

ProcessSpecName

Defines the underlying business process specification.

ProcessSpecUUID

Unique identifier of the BPSS ProcessSpecification.

ReceiverRoutingIdType

The ebXML party ID type of the message receiver.

RefToMessageId

When present, this must have a MessageId value of an earlier message to which this message relates. If there is no related earlier message, this element must not be used.

SenderRoutingIdType

The ebXML party ID type of the message sender.

Service

Identifies the collaborative business process. This can be bound to a service in back-end integration.

ServiceType

An option of the Service element, it is required only when the trading partners require a type to properly identify the service. If the Service is not a URI, a type must be specified.

SOAPAction

Identifies the action of the SOAP message header.

SyncReply

Indicates whether a response is used (such as a message level acknowledgment of a message for reliable messaging purposes). For example, instead of returning an HTTP code of 200 indicating success of a received ebXML message and then opening a new connection to send the message level acknowledgment back, the message level acknowledgment is sent over the same HTTP connection. The value of SyncReply can be **true** or **false**.

TimeToLive

If used, this indicates the time, expressed as universal time, that a message must be delivered. This attribute is optional.

ToRole

The role of the receiving party. Indicates the role the party is playing in the business transaction, such as the seller role.

MMD only

The following ebXML meta-data elements are used only in message meta-data documents.

PayloadLocation

The directory path to the document.

PayloadLocationType

The path and file name of the payload.

PayloadMimeContentId

The MIME type of the payload.

PayloadMimeContentType

An identifier of the payload content. For example, application/xml.

Required, optional meta-data for integration

The following table lists required and optional meta-data elements for all integration methods. The meta-data elements are defined in ebXML meta-data descriptions on page 447.

Element Name	Usage
AckRequested	Optional
AckSignatureRequested	Optional
Action	Optional
ConversationId	Optional
CPAId	Optional
DuplicateElimination	Optional
FromRole	Required
MessageId	Optional
PayloadLocation	MMD only
PayloadLocationType	MMD only
PayloadMimeContentId	MMD only
PayloadMimeContentType	MMD only
ProcessSpecName	Optional
ProcessSpecUUID	Optional
RefToMessageId	Optional

Element Name	Usage
Service	Optional but recommended for reliable performance
ServiceType	Required only if Service is specified and it is not a URI.
TimeToLive	Optional
ToRole	Required

Message meta-data documents

In file system integration with a back-end system, the trading engine supports ebXML business processes using message meta-data documents (MMDs) as the interface between it and the back-end. The MMDs are XML documents that point to an ebXML document on a file system and contain information the trading engine uses to process documents.

MMDs are used only with file system integration, although the same type of meta-data are transported when integration transports such as JMS or web services API are used.

MMD example

The following is an example of an MMD associated with an ebXML document. The trading engine generates MMDs for the ebXML documents that it sends to a back-end system. Your back-end system must generate the MMDs for ebXML documents the trading engine retrieves from integration. The meta-data elements are defined in ebXML message meta-data on page 446.

```
<?xml version="1.0" encoding="UTF-8"?>
<MessageMetadataDocument documentId="Test_E1" protocol="ebXML"
protocolVersion="2.0">
  <Metadata name="From" type="string">turpinsoft</Metadata>
  <Metadata name="FromRole" type="string">Test_E1</Metadata>
  <Metadata name="To" type="string">cyclone4</Metadata>
  <Metadata name="ToRole" type="string">Test_E1</Metadata>
  <Metadata name="Service" type="string">FileTransfer</Metadata>
  <Metadata name="Action" type="string">AsyncAck_E1</Metadata>
  <Metadata name="CPAId">urn:Cyclone4-turpinsoft</Metadata>
  <Metadata name="SyncReply">false</Metadata>
  <Metadata name="AckRequested">true</Metadata>
  <Metadata name="AckSignatureRequested">false</Metadata>
  <Metadata name="DuplicateElimination">false</Metadata>
  <MessagePayloads>
<Payload id="ID1234">
  <MimeContentId>TestE1</MimeContentId>
  <MimeContentType>application/xml</MimeContentType>
```

```
        <Location
          type="filePath">C:\dev\ebxml\payloads\smallxmlPO.xml
        </Location>
    </Payload>
    </MessagePayloads>
    </MessageMetadataDocument>
```

Using an MMD to ping a partner

Ping is an optional service that lets one message service handler determine whether another MSH is operating.

You can send a ping MMD to check network connectivity and the operational status of your and your partner’s systems. The receiver sends a pong response if all is well.

In Message tracker a ping is reported as a payload and a pong as a receipt. You can confirm a ping or pong by viewing the message contents.

Example of ping MMD

The following is an example of a ping MMD. Replace the placeholder values in this example with your sender and receiver information. Copy the MMD to a community file system integration pickup directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<MessageMetadataDocument documentId="PingMMD" protocol="ebXML"
protocolVersion="2.0">
  <Metadata name="From" type="string">PartnerA</Metadata>
  <Metadata name="To" type="string">PartnerB</Metadata>
  <Metadata name="Service">urn:oasis:names:tc:ebxml-msg:service</Metadata>
  <Metadata name="Action" type="string">Ping</Metadata>
  <Metadata name="CPAId">PartnerA-PartnerB-cpa-1</Metadata>
</MessageMetadataDocument>
```

Example of ping-pong messages

The following are examples of a ping message from one party and the pong response from the other party. These examples are packaged in SOAP envelopes.

Ping

The first example is the ping message sent by PartnerA.

```
Content-Type: text/xml
SOAPAction: "ebXML"
Content-Length: 1637

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
```

```
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/ http://
www.oasis-
open.org/committees/ebxml-msg/schema/envelope.xsd">
  <soap:Header xmlns:eb="http://www.oasis-open.org/committees/ebxml-msg/
schema/msg-header-
2_0.xsd" xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-
msg/schema/msg-header-
2_0.xsd http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-
2_0.xsd">
    <eb:MessageHeader eb:id="ID65722355511310407858 75coronation"
eb:version="2.0"
soap:mustUnderstand="1">
      <eb:From>
        <eb:PartyId eb:type="string">PartnerA</eb:PartyId>
      </eb:From>
      <eb:To>
        <eb:PartyId eb:type="string">PartnerB</eb:PartyId>
      </eb:To>
      <eb:CPAId>PartnerA-PartnerB-cpa-1</eb:CPAId>
      <eb:ConversationId>ab102b17-4724-4ecb-8572-8dc050a0f1a7</
eb:ConversationId>
      <eb:Service>urn:oasis:names:tc:ebxml-msg:service</eb:Service>
      <eb:Action>Ping</eb:Action>
      <eb:MessageData>
        <eb:MessageId>M1131040785868.792@coronation7786261718245588383</
eb:MessageId>
        <eb:Timestamp>2005-11-03T17:59:45.868Z</eb:Timestamp>
      </eb:MessageData>
    </eb:MessageHeader>
  </soap:Header>
  <soap:Body xmlns:eb="http://www.oasis-open.org/committees/ebxml-msg/
schema/msg-header-2_0.xsd"
xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-msg/schema/
msg-header-2_0.xsd
http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"/>
</soap:Envelope>
```

Pong

The second example is the pong reply from PartnerB.

```
POST http://PartnerA:4080/exchange/PartnerA HTTP/1.1
Content-Type: text/xml
SOAPAction: "ebXML"
User-Agent: haboob/5.3.3.0.6 build-1552
Host: coronation:4080
Connection: close
Content-Length: 1722
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/ http://
www.oasis-
open.org/committees/ebxml-msg/schema/envelope.xsd">
  <soap:Header xmlns:eb="http://www.oasis-open.org/committees/ebxml-msg/
schema/msg-header-
2_0.xsd" xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-
msg/schema/msg-header-
```

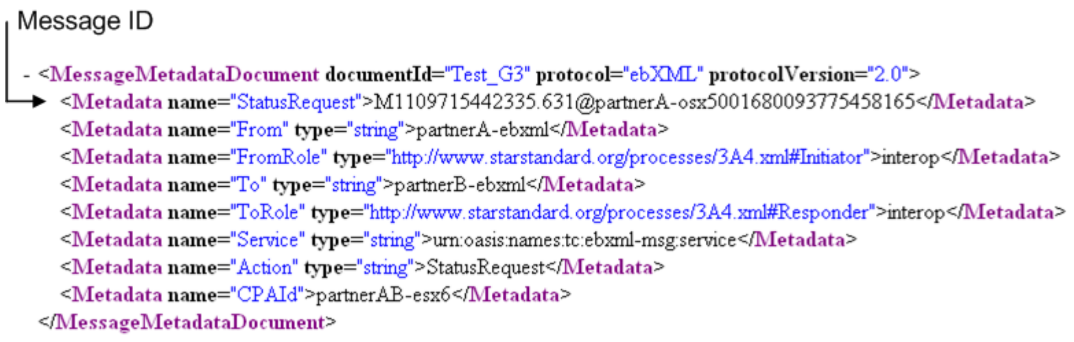
```
2_0.xsd http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-
2_0.xsd">
    <eb:MessageHeader eb:id="ID26808770113104058092 9gnaraloo"
eb:version="2.0"
soap:mustUnderstand="1">
      <eb:From>
        <eb:PartyId eb:type="string">PartnerB</eb:PartyId>
      </eb:From>
      <eb:To>
        <eb:PartyId eb:type="string">PartnerA</eb:PartyId>
      </eb:To>
      <eb:CPAId>PartnerA-PartnerB-cpa-1</eb:CPAId>
      <eb:ConversationId>ab102b17-4724-4ecb-8572-8dc050a0f1a7</
eb:ConversationId>
      <eb:Service>urn:oasis:names:tc:ebxml-msg:service</eb:Service>
      <eb:Action>Pong</eb:Action>
      <eb:MessageData>
        <eb:MessageId>M1131040580919.411688@gnaraloo8174619434348129230</
eb:MessageId>
        <eb:Timestamp>2005-11-03T17:56:20.919Z</eb:Timestamp>

<eb:RefToMessageId>M1131040785868.792@coronation7786261718245588383</
eb:RefToMessageId>
      </eb:MessageData>
    </eb:MessageHeader>
  </soap:Header>
  <soap:Body xmlns:eb="http://www.oasis-open.org/committees/ebxml-msg/
schema/msg-header-2_0.xsd"
xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-msg/schema/
msg-header-2_0.xsd
http://www.oasis-open.org/committees/ebxml-msg/schema/msg-header-2_0.xsd"/>
</soap:Envelope>
```

Using an MMD for a status request

You can use an MMD to ask a partner about the status of a previously sent message using the StatusRequest action. For instance, you could use this after reliable messaging retries in the CPA have been exhausted, but you want to check whether the partner received the payload.

Figure 138 is an example of a StatusRequest MMD. To use this MMD, you must provide an ebXML message ID as the value of the StatusRequest element near the top of the MMD. You also must provide your own values for From, To and CPAId.



```
- <MessageMetadataDocument documentId="Test_G3" protocol="ebXML" protocolVersion="2.0">
    <Metadata name="StatusRequest">M1109715442335.631@partnerA-osx5001680093775458165</Metadata>
    <Metadata name="From" type="string">partnerA-ebxml</Metadata>
    <Metadata name="FromRole" type="http://www.starstandard.org/processes/3A4.xml#Initiator">interop</Metadata>
    <Metadata name="To" type="string">partnerB-ebxml</Metadata>
    <Metadata name="ToRole" type="http://www.starstandard.org/processes/3A4.xml#Responder">interop</Metadata>
    <Metadata name="Service" type="string">urn:oasis:names:tc:ebxml-msg:service</Metadata>
    <Metadata name="Action" type="string">StatusRequest</Metadata>
    <Metadata name="CPAId">partnerAB-esx6</Metadata>
  </MessageMetadataDocument>
```

Figure 138. StatusRequest MMD

Supported trading transports

The supported transports for exchanging ebXML messages over the Internet are HTTP and e-mail (SMTP). When you configure a community to use one of these transports for receiving messages from partners, choose the ebXML message protocol in The delivery exchange wizard (see The delivery exchange wizard on page 189). Then select HTTP or SMTP as the transport. Follow the prompts to set up a transport.

Note: If you use embedded HTTP with client authentication (clients must use SSL to connect to this server), the community default SSL client authentication certificate is used, not the SSL certificate defined in the CPA. For more about SSL, see SSL authentication on page 316.

For more information about trading transports, see Trading delivery exchanges on page 183 and Adding transports on page 188.

Supported integration transports

A community can use any of the available transports for integration pickup and delivery. An integration pickup is when the trading engine retrieves a message from a back-end system. An integration delivery is when the trading engine sends a message received from a partner to a back-end system.

When choosing integration transports, decide what ebXML message meta-data will accompany the payloads.

The following topics describe the pickup and delivery integration options.

For more information about integration transports, see Integration exchanges on page 185 and Adding transports on page 188.

Integration pickup options

The following are the integration pickup options and meta-data sources for outbound documents.

The back-end or middleware generates meta-data

A back-end system or middleware can generate meta-data and forward the data to the trading engine. Middleware must understand meta-data and payloads. Meta-data picked up with ebXML payloads specifies, among other things, the CPAs to use for processing outbound messages to partners. The listed integration pickup transports must forward certain meta-data elements, as described in Required, optional meta-data for integration on page 451.

File system

The meta-data is contained in message meta-data documents (MMDs). The MMD contains a pointer to the location of the outbound document to be sent to the partner. See Message meta-data documents on page 452.

JMS

The meta-data must be set as JMS properties. The meta-data are attached to the BytesMessage containing the payload.

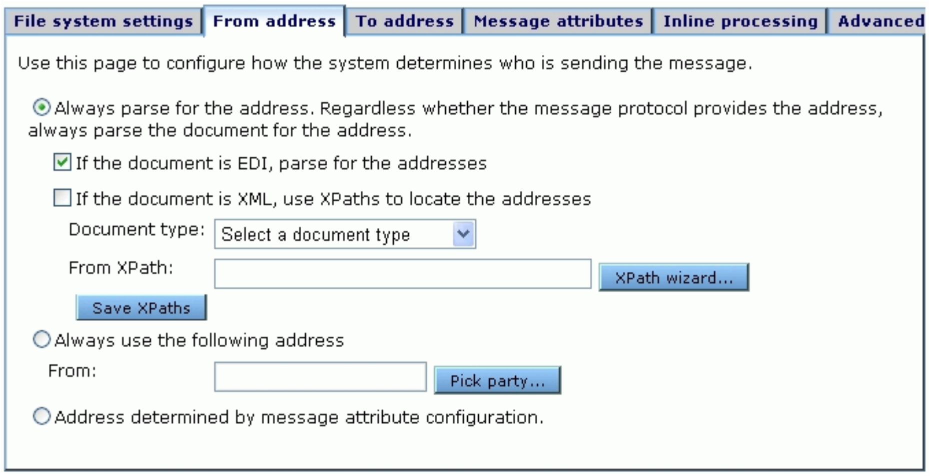
Web services API server

The meta-data are forwarded using the MessageService class. See MessageService.wsdl in [install directory]\[build number\conf. This WSDL defines how meta-data are set. For more information about this transport, see Web services API transport on page 221.

Using fixed meta-data

The simplest way to avoid having a back-end system produce meta-data for payloads is to specify the “from” and “to” address in the integration pickup transport and let the trading engine determine the other information it needs based on the community CPA. For file system integration, this method eliminates the need to use MMDs.

The only meta-data items you need to specify are “from” or SenderRoutingId(type) and “to” or ReceiverRoutingId(type) in the user interface. Use the **From address** or **To address** pages in the delivery exchange wizard or the **From address** and **To address** tabs on the exchange maintenance page.



Use this page to configure how the system determines who is sending the message.

● Always parse for the address. Regardless whether the message protocol provides the address, always parse the document for the address.

✓ If the document is EDI, parse for the addresses

☐ If the document is XML, use XPaths to locate the addresses

Document type: Select a document type

From XPath:

○ Always use the following address

From:

○ Address determined by message attribute configuration.

Figure 139. From address tab for integration pickup exchange

Figure 139 shows the **From address** tab for an integration pickup exchange. Select **Always use the following address** and click **Pick party** to select the sending party. Use the **To address** tab to make the same selections for the receiving party.

If you want to use this method, there are the following limitations:

- If a CPAId meta-data item is not specified, the trading engine looks up the default CPA for the given sender and receiver. The default CPA is the first in the list of imported CPAs for a community, if the sender and receiver have more than one.
- If the ToRole and FromRole meta-data items are not specified, the trading engine looks up the default Role for the sender party (FromRole). From this the trading engine determines the ToRole and the Service. The default Role is the first in the list of roles, so this should be used with a CPA with only one Role.
- If the Action is not specified, the trading engine uses the default CanSend action in the FromRole.

If the CPA is complex with many Roles and Actions or a community has more than one CPA, the “from” and “to” static meta-data method is not recommended.

An alternative to setting up only the “from” and “to” address is to add more static meta-data. With this method, rather than having a back-end system generate message meta-data, you can configure any integration pickup transport to attach to every message meta-data that you have defined. For configuration details see Fixed message attributes on page 297

For a list of the minimum name-value pairs, see Required, optional meta-data for integration on page 451.

Delivery integration options

You can use any of the available integration delivery transports to send messages received from partners to a back-end system. If, however, you want to attach message meta-data to payloads, there are three transport options.

File system with message meta-data

This integration delivery transport works just like the file system transport, but triggers the trading engine to produce an MMD and forward it with the inbound message to the back-end file system. The MMD contains all meta-data associated with the document received from the partner. See Message meta-data documents on page 452.

Web services API client

A client routes messages received from partners to a back-end server. The trading engine produces message meta-data to the back-end via the AsynchSend Java interface. The client sets a MetadataItem object on the API Message object for each meta-data item associated with the document received from the partner.

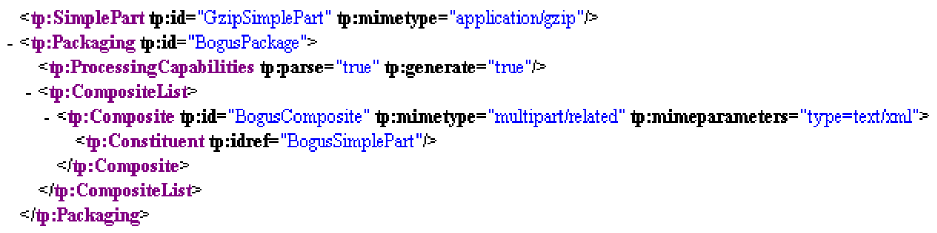
JMS

The message meta-data is attached to the BytesMessages containing the payload that is placed on the queue. The JMS exchange sets a JMS property on the BytesMessage for each meta-data item associated with the document received from the partner.

ebXML message compression

The trading engine supports GZIP compression of ebXML messages. To use it, your trading partner must support the same compression scheme.

To enable, you have to specify compression in the CPA Packaging element. Also, the CanSend and CanReceive elements must reference the defined Packaging element. Figure 140 is an example Packaging element.



```
  <tp:SimplePart tp:id="GzipSimplePart" tp:mimetype="application/gzip"/>
- <tp:Packaging tp:id="BogusPackage">
    <tp:ProcessingCapabilities tp:parse="true" tp:generate="true"/>
  - <tp:CompositeList>
    - <tp:Composite tp:id="BogusComposite" tp:mimetype="multipart/related" tp:mimeparameters="type=text/xml">
        <tp:Constituent tp:idref="BogusSimplePart"/>
      </tp:Composite>
    </tp:CompositeList>
  </tp:Packaging>
```

Figure 140. Elements in CPA to specify compression

Setting up a community for ebXML trading

Configuring a community profile for ebXML trading is much like setting up any community, but there are some notable differences. Primarily, the profile must be associated with a community- and partner-specific CPA for trading to occur. The CPA determines options for message packaging, requests for acknowledgments and compression, not the collaboration settings in the user interface. In addition, the community routing ID must have a specific format.

See The community profile on page 142 and Add a community on page 146 for details not specific to ebXML about setting up a a community profile.

Routing ID for ebXML

When setting up a community profile for ebXML, the routing ID must be set up in one of two ways:

- The routing ID (party ID) must be a Universal Resource Identifier (RFC 1630). For example, urn:myroutingid. When the routing ID is a URI, an ebXML party ID type is not necessary.

  or

- When the routing ID (party ID) is not a URI, enter an ebXML party ID type in addition to a unique routing ID. The ebXML party ID type can be any string (a URI, a DUNS number or something else) and the routing ID can be any unique string. For example: urn:mystring 123456789. The type value must match the PartyId /@type attribute value for the PartyInfo entry in the CPA for the community.

The user interface lets you set up routing IDs either way.

Moreover, if you trade with the same partner using both ebXML and any other message protocol, the community profile needs separate routing IDs for each protocol. The ebXML routing ID must not be the default. The default must be the routing ID used for the other message protocol.

The community profile and CPAs

The user interface lets you import complete CPAs as well as CPA templates and associate them with a community. Other than importing or deleting, however, tasks for managing CPAs are handled outside of this application. The exception is when you import a CPA template and have partners use the partner registration wizard. See Self-registration of ebXML partners on page 483.

When you import a complete CPA to the trading engine, several things occur. The CPA is associated with the importing community and the system creates a partner profile for the partner specified in the CPA. The system extracts from the CPA the public key certificate data and the URL or e-mail address for sending messages and adds them to the partner profile.

Your community profile is not affected when a CPA is imported. During the importing process, however, the trading engine checks whether the community keystore contains the corresponding private keys for all the signing and encrypting public keys in the PartyInfo section of the CPA.

Before a community can import a complete CPA, the community profile must be set up on the trading engine. The community name, routing ID, and certificate data must be the same in the profile and the CPA.

Once the community and partner profiles are configured and a CPA is associated with the community, message exchanges can begin.

Anatomy of a CPA

A Collaboration Protocol Agreement (CPA) is a complex XML document. A CPA is an agreement between two parties for exchanging e-commerce messages over the Internet. The CPA stipulates what type of messages are to be exchanged. The agreement also contains technical details for transporting messages via HTTP or SMTP and how messages are packaged (encrypted, signed and the digital envelope protocol).

Building a CPA document is a process that takes place outside of the user interface of Cyclone Activator. CPAs must conform to the Collaboration-Protocol Profile and Agreement Specification Version 2.0. The supported CPA schema is http://www.oasis-open.org/committees/ebxml-cppa/

schema/cpp-cpa-2_0.xsd. Creating and maintaining CPAs is a manual process unless aided with tools available commercially or developed in-house.

You must understand XML documents and the rules for CPAs to build CPAs. This documentation partially explains CPAs, but does not take the place of information available from authoritative sources such as www.ebxml.org or www.oasis-open.org. For complete information about CPAs, see Collaboration-Protocol Profile and Agreement Specification 2.0, available at http://www.oasis-open.org/committees/ebxml-cppa/documents/ebcpp-2.0.pdf (Adobe Acrobat required).

One CPA is required for each trading relationship, defined as your community and one partner. This one-to-one relationship means you need one CPA for each partner. For example, if your community has 15 partners whose preferred protocol is ebXML, you need 15 CPAs.

Cyclone Activator performs minimal checks and validations upon importing a CPA. If the trading engine cannot add a profile for the partner named in the CPA, the import fails. However, if the import succeeds, there could be omissions or errors in the CPA that may result in a trading failure. If trading fails for CPA-related issues, you must correct the CPA and import it again to the trading engine, overwriting the previous CPA.

Common CPA elements

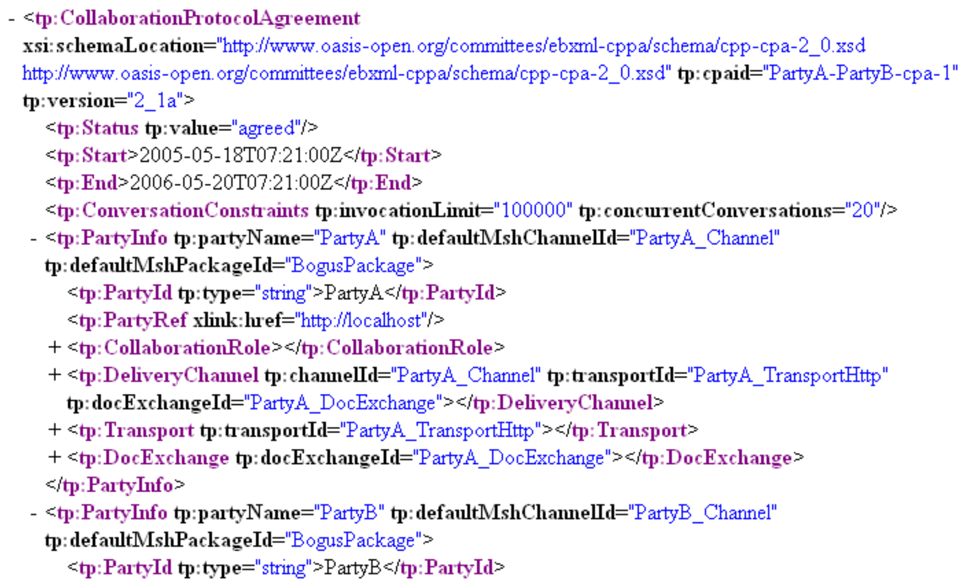
The following describes elements common to CPAs and Collaboration Protocol Profiles (CPPs). CPAs and CPPs have similar structures. The primary difference is a CPA is for two parties, while a CPP is for a single party. Building a CPA can be a matter of merging two CPPs, as described in Collaboration-Protocol Profile and Agreement Specification Version 2.0.

In the following topics, groups of elements are described in top-down order for a basic CPA. The CPA cited in this example includes the bare minimum of elements for testing connectivity between partners. There are no settings for encryption, digital signing or SSL.

CollaborationProtocolAgreement and PartyInfo elements

The first element in a CPA is **CollaborationProtocolAgreement** (Figure 141). The two parties are represented by the two **PartyInfo** elements. In a CPA there are exactly two PartyInfo elements.

The PartyInfo element has a **Name** attribute, which specifies the name of the party. Also included is **defaultMSHChannelId**, which selects a **DeliveryChannel** to send MSH signals, such as acknowledgments, errors, ping or pong messages. PartyInfo also specifies the **PartyId** and a **type** attribute whose values are used in the ebXML message.



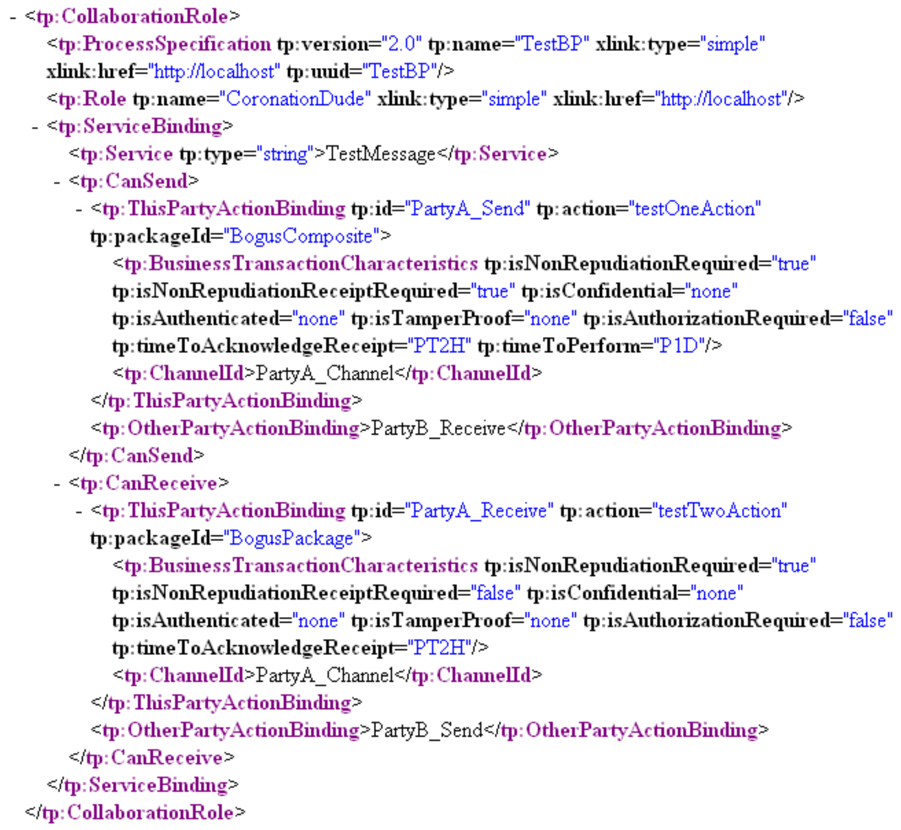
```
- <tp:CollaborationProtocolAgreement
  xsi:schemaLocation="http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd
  http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd" tp:cpaid="PartyA-PartyB-cpa-1"
  tp:version="2_1a">
    <tp:Status tp:value="agreed"/>
    <tp:Start>2005-05-18T07:21:00Z</tp:Start>
    <tp:End>2006-05-20T07:21:00Z</tp:End>
    <tp:ConversationConstraints tp:invocationLimit="100000" tp:concurrentConversations="20"/>
  - <tp:PartyInfo tp:partyName="PartyA" tp:defaultMshChannelId="PartyA_Channel"
    tp:defaultMshPackageId="BogusPackage">
      <tp:PartyId tp:type="string">PartyA</tp:PartyId>
      <tp:PartyRef xlink:href="http://localhost"/>
    + <tp:CollaborationRole></tp:CollaborationRole>
    + <tp:DeliveryChannel tp:channelId="PartyA_Channel" tp:transportId="PartyA_TransportHttp"
      tp:docExchangeId="PartyA_DocExchange"></tp:DeliveryChannel>
    + <tp:Transport tp:transportId="PartyA_TransportHttp"></tp:Transport>
    + <tp:DocExchange tp:docExchangeId="PartyA_DocExchange"></tp:DocExchange>
    </tp:PartyInfo>
  - <tp:PartyInfo tp:partyName="PartyB" tp:defaultMshChannelId="PartyB_Channel"
    tp:defaultMshPackageId="BogusPackage">
      <tp:PartyId tp:type="string">PartyB</tp:PartyId>
```

Figure 141. CollaborationProtocolAgreement section

CollaborationRole element

A CPA lists all the collaborative business processes agreed upon by the two parties. This is done through the **CollaborationRole** element (Figure 142). Each collaborative business process is represented by one CollaborationRole element. The PartyInfo includes all information for how to exchange the business document of the collaborative business process. Because this information is provided in each of the two PartyInfo elements, a CPA can be large and difficult to read.

The CollaborationRole states what documents are being sent and received.



```
- <tp:CollaborationRole>
    <tp:ProcessSpecification tp:version="2.0" tp:name="TestBP" xlink:type="simple"
    xlink:href="http://localhost" tp:uuid="TestBP"/>
    <tp:Role tp:name="CoronationDude" xlink:type="simple" xlink:href="http://localhost"/>
  - <tp:ServiceBinding>
      <tp:Service tp:type="string">TestMessage</tp:Service>
    - <tp:CanSend>
      - <tp:ThisPartyActionBinding tp:id="PartyA_Send" tp:action="testOneAction"
        tp:packageId="BogusComposite">
          <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="true"
          tp:isNonRepudiationReceiptRequired="true" tp:isConfidential="none"
          tp:isAuthenticated="none" tp:isTamperProof="none" tp:isAuthorizationRequired="false"
          tp:timeToAcknowledgeReceipt="PT2H" tp:timeToPerform="P1D"/>
          <tp:ChannelId>PartyA_Channel</tp:ChannelId>
        </tp:ThisPartyActionBinding>
        <tp:OtherPartyActionBinding>PartyB_Receive</tp:OtherPartyActionBinding>
      </tp:CanSend>
    - <tp:CanReceive>
      - <tp:ThisPartyActionBinding tp:id="PartyA_Receive" tp:action="testTwoAction"
        tp:packageId="BogusPackage">
          <tp:BusinessTransactionCharacteristics tp:isNonRepudiationRequired="true"
          tp:isNonRepudiationReceiptRequired="false" tp:isConfidential="none"
          tp:isAuthenticated="none" tp:isTamperProof="none" tp:isAuthorizationRequired="false"
          tp:timeToAcknowledgeReceipt="PT2H"/>
          <tp:ChannelId>PartyA_Channel</tp:ChannelId>
        </tp:ThisPartyActionBinding>
        <tp:OtherPartyActionBinding>PartyB_Send</tp:OtherPartyActionBinding>
      </tp:CanReceive>
    </tp:ServiceBinding>
  </tp:CollaborationRole>
```

Figure 142. CollaborationRole section

Within the CollaborationRole element, the **ProcessSpecification** element identifies the exact collaborative business process. It is the **Role** element following ProcessSpecification that binds the party represented by the PartyInfo element to an actual role of the collaborative business process.

In ebXML a collaborative business process is composed of choreographed business transactions. There are different types of transactions. A transaction can be a simple notification message or one that involves a series of documents triggered in request-response sequences. An instance of a collaborative business process is abstract because it does not specify parties but roles. The CPA is required to bind the parties who execute the roles of the collaborative business process.

The messages — the business documents exchanged throughout the collaborative business process — are listed in the **ServiceBinding** element, a child element of CollaborationRole.

The choreography information (the sequence of business documents exchanged throughout the collaborative business process) is omitted in the CPA. The CPA, however, does list each business document of the collaborative business process. The CPA also specifies the sender and receiver of a particular business document in the **CanSend** and **CanReceive** elements.

For each business document listed as CanSend for one party, the same document is listed as CanReceive for the other party. To make identifying documents easier, CanSend has a **ThisPartyActionBinding** element that corresponds to a CanReceive **OtherPartyActionBinding** element.

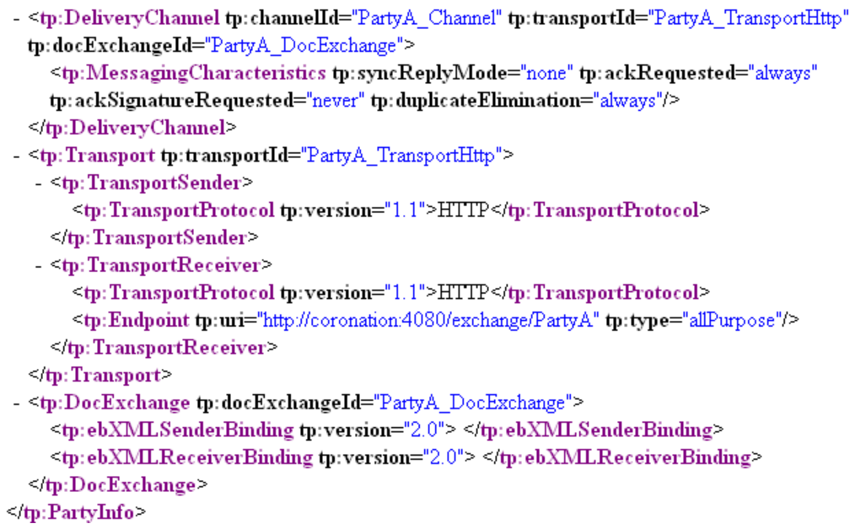
The ThisPartyActionBinding and OtherPartyActionBinding elements have an **action** attribute that identifies the business document. These elements also have a **packageId** attribute specifying how the messages are packaged.

The **BusinessTransactionCharacteristics** element, a child of the ThisPartyActionBinding element, includes attributes for message security settings. Cyclone Activator does not support this element or its attributes, but relies on security settings elsewhere in the CPA (NonRepudiation, ReliableMessaging, DigitalEnvelope). However, when you build a CPA, you must include the BusinessTransactionCharacteristics element and its attributes, as your partner's trading engine may require these settings.

Another ThisPartyActionBinding element is the **ChannelId** element, which references the DeliveryChannel element that specifies the technical information for the method for transporting the message.

DeliveryChannel element

The ThisPartyActionBinding element ChannelId references the **DeliveryChannel** element (Figure 143). Multiple ChannelId elements can reference the same DeliveryChannel. The DeliveryChannel element in turn references the **Transport** and **DocExchange** elements.



```
- <tp:DeliveryChannel tp:channelId="PartyA_Channel" tp:transportId="PartyA_TransportHttp"
  tp:docExchangeId="PartyA_DocExchange">
    <tp:MessagingCharacteristics tp:syncReplyMode="none" tp:ackRequested="always"
    tp:ackSignatureRequested="never" tp:duplicateElimination="always"/>
  </tp:DeliveryChannel>
- <tp:Transport tp:transportId="PartyA_TransportHttp">
  - <tp:TransportSender>
      <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
    </tp:TransportSender>
  - <tp:TransportReceiver>
      <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
      <tp:Endpoint tp:uri="http://coronation:4080/exchange/PartyA" tp:type="allPurpose"/>
    </tp:TransportReceiver>
  </tp:Transport>
- <tp:DocExchange tp:docExchangeId="PartyA_DocExchange">
    <tp:ebXMLSenderBinding tp:version="2.0"> </tp:ebXMLSenderBinding>
    <tp:ebXMLReceiverBinding tp:version="2.0"> </tp:ebXMLReceiverBinding>
  </tp:DocExchange>
</tp:PartyInfo>
```

Figure 143. DeliveryChannel, Transport, DocExchange elements

The DeliveryChannel element has a **MessagingCharacteristics** element that sets some collaboration settings, no matter the underlying transport protocol or document exchange settings. The MessagingCharacteristics element has a **syncReply** attribute to indicate whether the response should be an ebXML MSH acknowledgment or, in the case of the HTTP transport, a business document.

The DocExchange element in Figure 143 demonstrates that this CPA has no settings for reliable messaging or encryption. For examples of such settings, see Reliable messaging, non-repudiation and encryption on page 466.

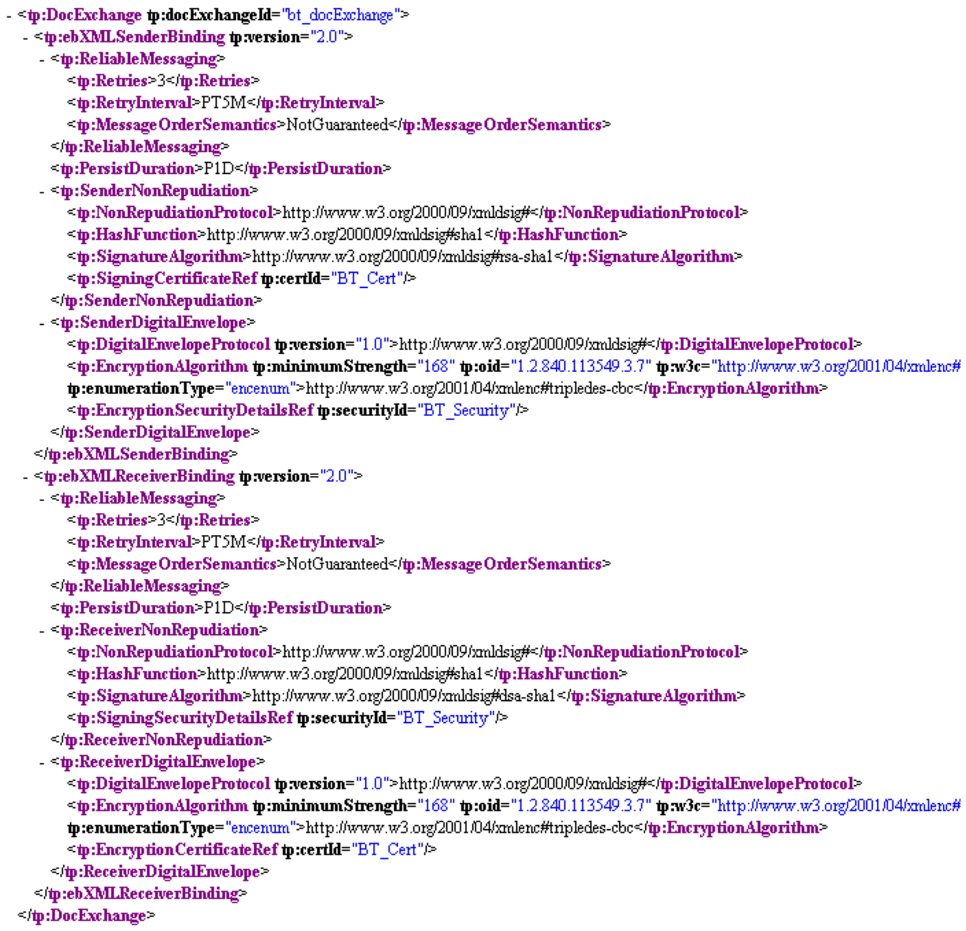
Note: When business documents are sent asynchronously a receiver is typically the server and the sender is typically the client.

A DeliveryChannel can be used for sending and receiving documents, meaning it can be referenced from the CanSend and CanReceive elements. Correspondingly, the Transport element includes a **TransportSender** and a **TransportReceiver**. Each element has a TransportProtocol element to identify the transport method (HTTP or SMTP). Following the **TransportReceiver** element, the **Endpoint** element provides the delivery address (for HTTP, a URL; for SMTP, an e-mail address).

Reliable messaging, non-repudiation and encryption

This topic explains elements under the DocExchange element that control reliable messaging and non-repudiation.

The DocExchange element (Figure 144) has sub-elements containing message security rules for sending and receiving messages. These elements are **ebXMLSenderBinding** and **ebXMLReceiverBinding**. The ebXMLSenderBinding of the first PartyInfo must be compatible with the ebXMLReceiverBinding of the second PartyInfo.



```
- <tp:DocExchange tp:docExchangeId="bt_docExchange">
  - <tp:ebXMLSenderBinding tp:version="2.0">
    - <tp:ReliableMessaging>
        <tp:Retries>3</tp:Retries>
        <tp:RetryInterval>PT5M</tp:RetryInterval>
        <tp:MessageOrderSemantics>NotGuaranteed</tp:MessageOrderSemantics>
      </tp:ReliableMessaging>
      <tp:PersistDuration>P1D</tp:PersistDuration>
    - <tp:SenderNonRepudiation>
        <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
        <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
        <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</tp:SignatureAlgorithm>
        <tp:SigningCertificateRef tp:certId="BT_Cert"/>
      </tp:SenderNonRepudiation>
    - <tp:SenderDigitalEnvelope>
        <tp:DigitalEnvelopeProtocol tp:version="1.0">http://www.w3.org/2000/09/xmldsig#</tp:DigitalEnvelopeProtocol>
        <tp:EncryptionAlgorithm tp:minimumStrength="168" tp:oid="1.2.840.113549.3.7" tp:w3c="http://www.w3.org/2001/04/xmlenc#
        tp:enumerationType="encenum">http://www.w3.org/2001/04/xmlenc#tripledes-cbc</tp:EncryptionAlgorithm>
        <tp:EncryptionSecurityDetailsRef tp:securityId="BT_Security"/>
      </tp:SenderDigitalEnvelope>
    </tp:ebXMLSenderBinding>
  - <tp:ebXMLReceiverBinding tp:version="2.0">
    - <tp:ReliableMessaging>
        <tp:Retries>3</tp:Retries>
        <tp:RetryInterval>PT5M</tp:RetryInterval>
        <tp:MessageOrderSemantics>NotGuaranteed</tp:MessageOrderSemantics>
      </tp:ReliableMessaging>
      <tp:PersistDuration>P1D</tp:PersistDuration>
    - <tp:ReceiverNonRepudiation>
        <tp:NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</tp:NonRepudiationProtocol>
        <tp:HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</tp:HashFunction>
        <tp:SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-sha1</tp:SignatureAlgorithm>
        <tp:SigningSecurityDetailsRef tp:securityId="BT_Security"/>
      </tp:ReceiverNonRepudiation>
    - <tp:ReceiverDigitalEnvelope>
        <tp:DigitalEnvelopeProtocol tp:version="1.0">http://www.w3.org/2000/09/xmldsig#</tp:DigitalEnvelopeProtocol>
        <tp:EncryptionAlgorithm tp:minimumStrength="168" tp:oid="1.2.840.113549.3.7" tp:w3c="http://www.w3.org/2001/04/xmlenc#
        tp:enumerationType="encenum">http://www.w3.org/2001/04/xmlenc#tripledes-cbc</tp:EncryptionAlgorithm>
        <tp:EncryptionCertificateRef tp:certId="BT_Cert"/>
      </tp:ReceiverDigitalEnvelope>
    </tp:ebXMLReceiverBinding>
  </tp:DocExchange>
```

Figure 144. Reliable messaging, non-repudiation and encryption

Both the sender and receiver binding elements have **ReliableMessaging** sub-elements for **NonRepudiation** and **DigitalEnvelope**.

ReliableMessaging has important child elements. **Retries** indicates how many times a business document must be sent if an acknowledgment has not been received from the partner. If retries are exhausted, the document is assigned a failed status. **RetryInterval** indicates the elapsed time between retries. If a signed or unsigned acknowledgment is requested and retries is set to **0**, RetryInterval is the time to wait before failing the document.

Another ReliableMessaging child element, **MessageOrderSemantics**, is not supported by Cyclone Activator.

Cyclone Activator does not use the **PersistDuration** element.

Important child elements of NonRepudiation are:

- **NonRepudiationProtocol** identifies the protocol to use. XML Digital Signatures (XMLDSIG) is the defacto protocol. The schema for XML signatures is at:

  http://www.w3.org/2000/09/xmldsig#

- **HashFunction** must be **sha1**. This is the only one Cyclone Activator supports. This is the algorithm for the digital signature hash.

  http://www.w3.org/2000/09/xmldsig#sha1

- **SignatureAlgorithm** indicates algorithms such as **dsa-sha1** and **rsa-sha1**. This is for signing the signature hash with the sender's certificate.

  The following schemas are supported:

  http://www.w3.org/2000/09/xmldsig#dsa-sha1
  http://www.w3.org/2000/09/xmldsig#rsa-sha1

  For a self-signed certificate generated by Cyclone Activator, only rsa-sha1 is supported because the certficate uses RSA encryption.

- SenderNonRepudiation includes **SignatureCertificateRef**, which references the certificate the sending party uses to sign business documents.

- ReceiverNonRepudiation includes **SigningSecurityDetailsRef**, which references a list of trusted certificates. When a signed message is received, it must have been signed by a trusted certificate in this list.

Important child elements of DigitalEnvelope are:

- **DigitalEnvelopeProtocol** identifies the protocol, typically **XMLENC**.

- **EncryptionAlgorithm** specifiies the encryption algorithm.

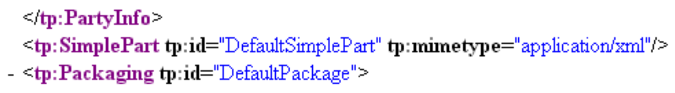
- SenderDigitalEnvelope includes **EncryptionSecurityDetailsRef**. When the sender encrypts a message with the partner's certificate, the certificate must be digitally signed by one of the trusted certificates in this list.

- ReceiverDigitalEnvelope includes EncryptionCertificateRef, which references the certificate the sender used to encrypt the message.

When certificates are used for signing or encryption, the public keys are published as XMLDSIG **KeyInfo** elements in the CPA. The **SecurityDetails** element and its child element **TrustAnchors** list trused certificates.

SimplePart and Packaging elements

CPAs specify how messages are packaged with the **SimplePart** and **Packaging** elements (Figure 145). Cyclone Activator ignores these elements, but packages messages according to ebXML messaging specifications. Although not supported, you must include these elements in CPAs to accommodate partners and comply with CPA standards.



```
  </tp:PartyInfo>
  <tp:SimplePart tp:id="DefaultSimplePart" tp:mimetype="application/xml"/>
- <tp:Packaging tp:id="DefaultPackage">
```

Figure 145. SimplePart and Packaging elements

SimplePart provides information about the individual parts of an ebXML message, such as the message headers, MSH signals, business process signals, as well as the individual business documents.

The Packaging element composes the individual SimplePart’s together. For example, an ebXML message header and a business document SimplePart. The Packaging element further specifies various MIME related settings.

Extracting the KeyInfo element for a CPA

A tool is available for exporting public key information from a certificate. The KeyInfo element information then can be copied to a CPA.

Export the community or partner certificate to a file. The file must have an extension of .cer, .p7b or .p7c. The tool for extracting the KeyInfo element information is keyInfoWriter.cmd (Windows) or keyInfoWriter (UNIX). The tool is in [install directory]\[build number]\tools.

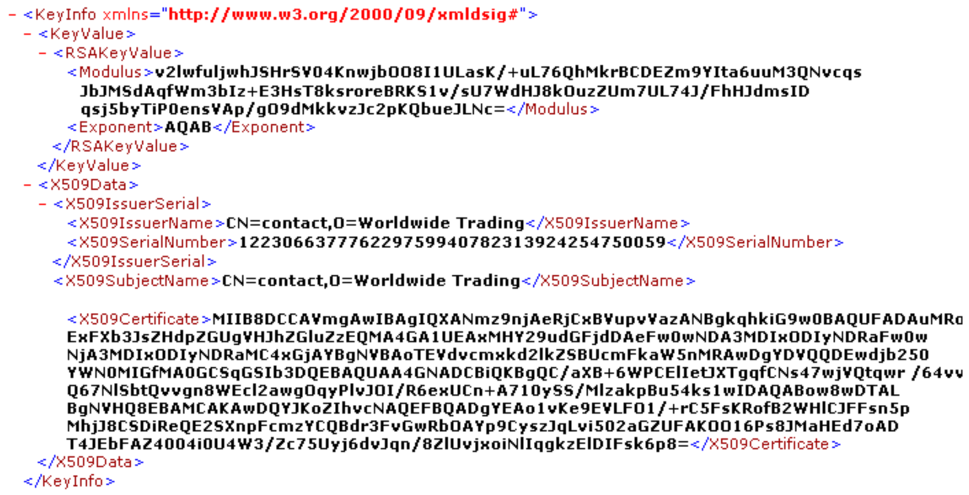
This command line tool must be run from the tools directory in the following format:

```
keyInfoWriter [path of certificate file] [path of XML output file]
```

The following is an example use of the tool:

```
keyInfoWriter c:\certificates\worldwide.p7b c:\certificates\
worldwide_keyinfo.xml
```

Figure 146 illustrates the example worldwide_keyinfo.xml output file. This is only a partial view of the file. Some lines are longer than the width of the illustration and are not shown.



```
- <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
  - <KeyValue>
    - <RSAKeyValue>
        <Modulus>v2lwfuljwhJSHrSV04KnwjbOO8I1ULasK/+uL76QhMkrBCDEZm9YIta6uuM3QNvcqs
         JbJMSdAqfWm3bIz+E3HsT8ksroreBRKS1v/sU7WdHJ8kOuzZUm7UL74J/FhHJdmsID
         qsj5byTiP0ensVAp/gO9dMkkvzJc2pKQbueJLNc=</Modulus>
        <Exponent>AQAB</Exponent>
      </RSAKeyValue>
    </KeyValue>
  - <X509Data>
    - <X509IssuerSerial>
        <X509IssuerName>CN=contact,O=Worldwide Trading</X509IssuerName>
        <X509SerialNumber>122306637776229759940782313924254750059</X509SerialNumber>
      </X509IssuerSerial>
      <X509SubjectName>CN=contact,O=Worldwide Trading</X509SubjectName>

        <X509Certificate>MIIB8DCCAVmgAwIBAgIQXANmz9njAeRjCxBVupvVazANBgkqhkiG9w0BAQUFADAuMRo
        ExFXb3JsZHdpZGUgVHJhZGluZzEQMA4GA1UEAxMHY29udGFjdDAeFw0wNDA3MDIxODIyNDRaFw0w
        NjA3MDIxODIyNDRaMC4xGjAYBgNVBAoTEVdvcmxkd2lkZSBUcmFkaW5nMRAwDgYDVQQDEwdjb250
        YWN0MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC/aXB+6WPCElIetJXTgqfCNs47wjVQtqwr /64vv
        Q67NlSbtQvvgn8WEcl2awgOqyPlvJOI/R6exUCn+A710ySS/MlzakpBu54ks1wIDAQABow8wDTAL
        BgNVHQ8EBAMCAKAwDQYJKoZIhvcNAQEFBQADgYEAo1vKe9EVLFO1/+rC5FsKRofB2WHlCJFFsn5p
        MhjJ8CSDiReQE2SXnpFcmzYCQBdr3FvGwRbOAYp9CyszJqLvi502aGZUFAKOO16Ps8JMaHEd7oAD
        T4JEbFAZ4004i0U4W3/Zc75Uyj6dvJqn/8ZlUvjxoiNlIqgkzElDIFsk6p8=</X509Certificate>
    </X509Data>
  </KeyInfo>
```

Figure 146. Example of XML file with KeyInfo element data

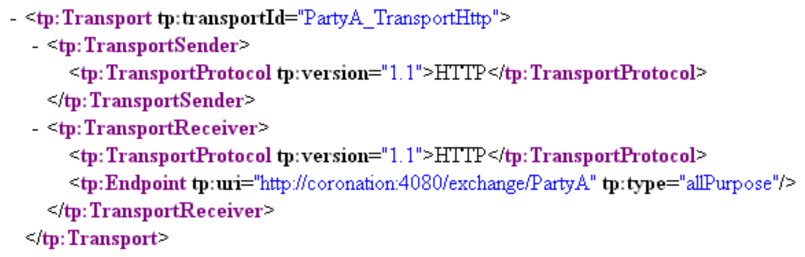
CPA use case examples

The following are snippets of CPAs showing settings for:

- HTTP transport
- HTTPS transport
- SMTP transport
- Trading without security
- Reliable messaging
- XMLENC encryption
- S/MIME encryption
- Digital signature

HTTP transport

Figure 147 shows the use of HTTP as the transport protocol.

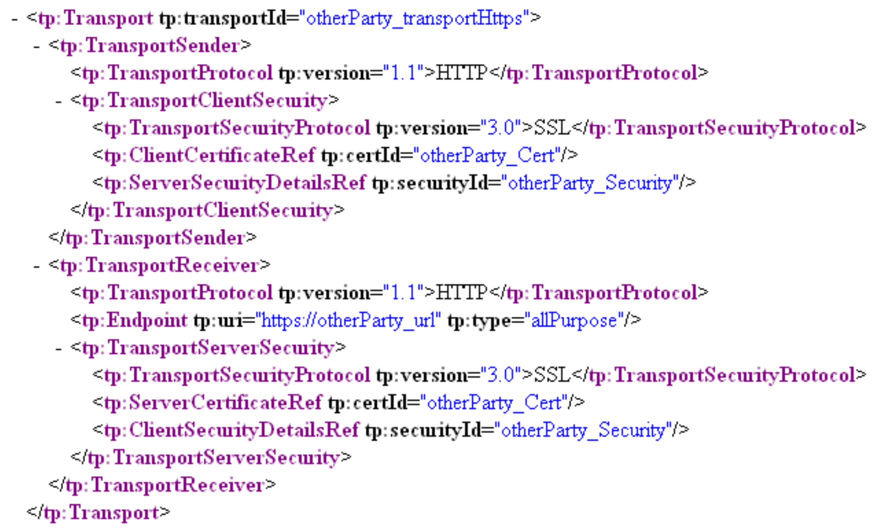


```
- <tp:Transport tp:transportId="PartyA_TransportHttp">
  - <tp:TransportSender>
      <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
    </tp:TransportSender>
  - <tp:TransportReceiver>
      <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
      <tp:Endpoint tp:uri="http://coronation:4080/exchange/PartyA" tp:type="allPurpose"/>
    </tp:TransportReceiver>
  </tp:Transport>
```

Figure 147. HTTP transport

HTTPS transport

Figure 148 shows the use of HTTPS (HTTP over SSL) as the transport protocol. The Endpoint element contains the HTTPS URL.

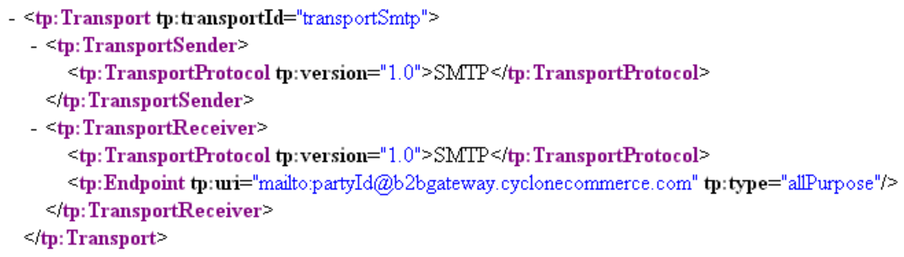


```
- <tp:Transport tp:transportId="otherParty_transportHttps">
  - <tp:TransportSender>
      <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
    - <tp:TransportClientSecurity>
        <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
        <tp:ClientCertificateRef tp:certId="otherParty_Cert"/>
        <tp:ServerSecurityDetailsRef tp:securityId="otherParty_Security"/>
      </tp:TransportClientSecurity>
    </tp:TransportSender>
  - <tp:TransportReceiver>
      <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
      <tp:Endpoint tp:uri="https://otherParty_url" tp:type="allPurpose"/>
    - <tp:TransportServerSecurity>
        <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:TransportSecurityProtocol>
        <tp:ServerCertificateRef tp:certId="otherParty_Cert"/>
        <tp:ClientSecurityDetailsRef tp:securityId="otherParty_Security"/>
      </tp:TransportServerSecurity>
    </tp:TransportReceiver>
  </tp:Transport>
```

Figure 148. HTTPS transport

SMTP transport

Figure 149 shows the use of SMTP as the transport protocol. In the Endpoint element, note the use of the **mailto** prefix before the e-mail address.

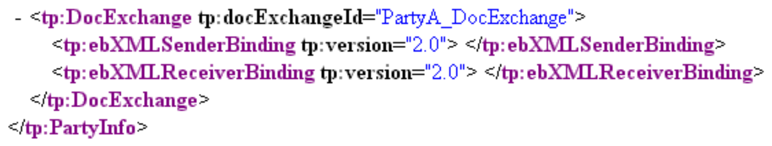


```
- <tp:Transport tp:transportId="transportSmtp">
  - <tp:TransportSender>
      <tp:TransportProtocol tp:version="1.0">SMTP</tp:TransportProtocol>
    </tp:TransportSender>
  - <tp:TransportReceiver>
      <tp:TransportProtocol tp:version="1.0">SMTP</tp:TransportProtocol>
      <tp:Endpoint tp:uri="mailto:partyId@b2bgateway.cyclonecommerce.com" tp:type="allPurpose"/>
    </tp:TransportReceiver>
  </tp:Transport>
```

Figure 149. SMTP transport

Trading without security

Figure 150 shows a DocExchange element without any security enabled (that is, no reliable messaging, encryption or digital signature.

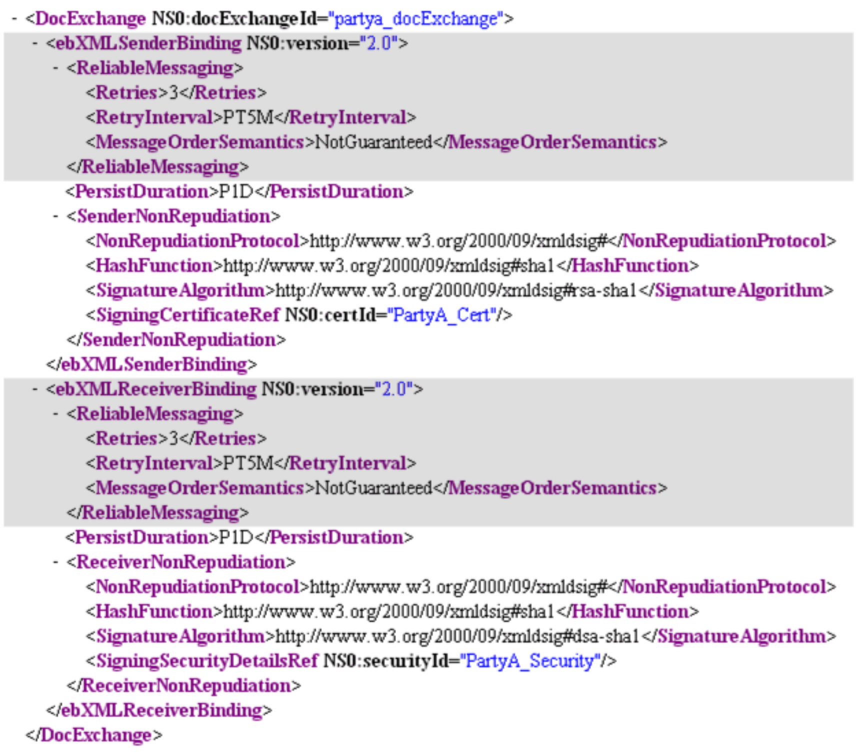


```
 - <tp:DocExchange tp:docExchangeId="PartyA_DocExchange">
     <tp:ebXMLSenderBinding tp:version="2.0"> </tp:ebXMLSenderBinding>
     <tp:ebXMLReceiverBinding tp:version="2.0"> </tp:ebXMLReceiverBinding>
   </tp:DocExchange>
</tp:PartyInfo>
```

Figure 150. DocExchange without security

Reliable messaging

The shaded areas of Figure 151 show reliable messaging.



```
- <DocExchange NS0:docExchangeId="partya_docExchange">
  - <ebXMLSenderBinding NS0:version="2.0">
    - <ReliableMessaging>
        <Retries>3</Retries>
        <RetryInterval>PT5M</RetryInterval>
        <MessageOrderSemantics>NotGuaranteed</MessageOrderSemantics>
      </ReliableMessaging>
      <PersistDuration>P1D</PersistDuration>
    - <SenderNonRepudiation>
        <NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</NonRepudiationProtocol>
        <HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</HashFunction>
        <SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</SignatureAlgorithm>
        <SigningCertificateRef NS0:certId="PartyA_Cert"/>
      </SenderNonRepudiation>
    </ebXMLSenderBinding>
  - <ebXMLReceiverBinding NS0:version="2.0">
    - <ReliableMessaging>
        <Retries>3</Retries>
        <RetryInterval>PT5M</RetryInterval>
        <MessageOrderSemantics>NotGuaranteed</MessageOrderSemantics>
      </ReliableMessaging>
      <PersistDuration>P1D</PersistDuration>
    - <ReceiverNonRepudiation>
        <NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</NonRepudiationProtocol>
        <HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</HashFunction>
        <SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-sha1</SignatureAlgorithm>
        <SigningSecurityDetailsRef NS0:securityId="PartyA_Security"/>
      </ReceiverNonRepudiation>
    </ebXMLReceiverBinding>
  </DocExchange>
```

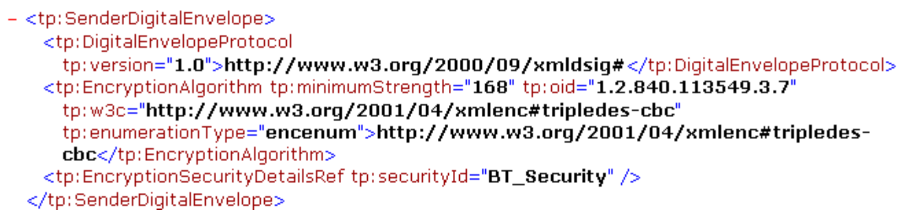
Figure 151. Reliable messaging

XMLENC encryption

Figure 152 and Figure 153 show XMLENC encryption.

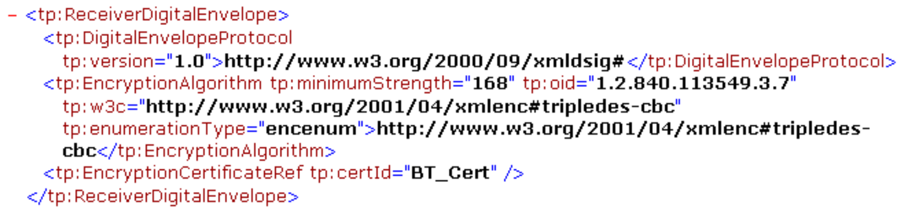
Supported XMLENC algorithms are:

- http://www.w3.org/2001/04/xmlenc#tripledes-cbc
- http://www.w3.org/2001/04/xmlenc#aes128-cbc
- http://www.w3.org/2001/04/xmlenc#aes192-cbc
- http://www.w3.org/2001/04/xmlenc#aes256-cbc



```
- <tp:SenderDigitalEnvelope>
    <tp:DigitalEnvelopeProtocol
      tp:version="1.0">http://www.w3.org/2000/09/xmldsig#</tp:DigitalEnvelopeProtocol>
    <tp:EncryptionAlgorithm tp:minimumStrength="168" tp:oid="1.2.840.113549.3.7"
      tp:w3c="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"
      tp:enumerationType="encenum">http://www.w3.org/2001/04/xmlenc#tripledes-
      cbc</tp:EncryptionAlgorithm>
    <tp:EncryptionSecurityDetailsRef tp:securityId="BT_Security" />
  </tp:SenderDigitalEnvelope>
```

Figure 152. SenderDigitalEnvelope with XMLENC

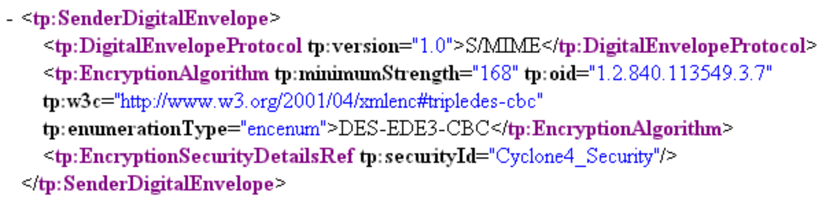


```
- <tp:ReceiverDigitalEnvelope>
    <tp:DigitalEnvelopeProtocol
      tp:version="1.0">http://www.w3.org/2000/09/xmldsig#</tp:DigitalEnvelopeProtocol>
    <tp:EncryptionAlgorithm tp:minimumStrength="168" tp:oid="1.2.840.113549.3.7"
      tp:w3c="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"
      tp:enumerationType="encenum">http://www.w3.org/2001/04/xmlenc#tripledes-
      cbc</tp:EncryptionAlgorithm>
    <tp:EncryptionCertificateRef tp:certId="BT_Cert" />
  </tp:ReceiverDigitalEnvelope>
```

Figure 153. ReceiverDigitalEnvelope with XMLENC

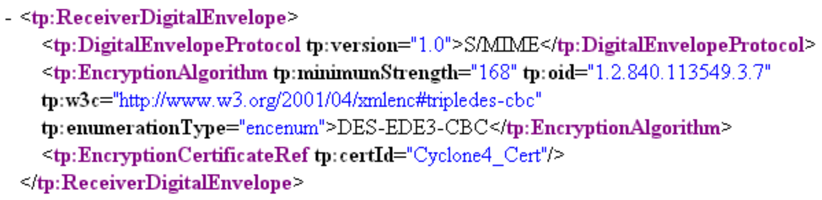
S/MIME encryption

Figure 154 and Figure 155 show S/MIME encryption.



```
- <tp:SenderDigitalEnvelope>
    <tp:DigitalEnvelopeProtocol tp:version="1.0">S/MIME</tp:DigitalEnvelopeProtocol>
    <tp:EncryptionAlgorithm tp:minimumStrength="168" tp:oid="1.2.840.113549.3.7"
    tp:w3c="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"
    tp:enumerationType="encenum">DES-EDE3-CBC</tp:EncryptionAlgorithm>
    <tp:EncryptionSecurityDetailsRef tp:securityId="Cyclone4_Security"/>
  </tp:SenderDigitalEnvelope>
```

Figure 154. SenderDigitalEnvelope with S/MIME

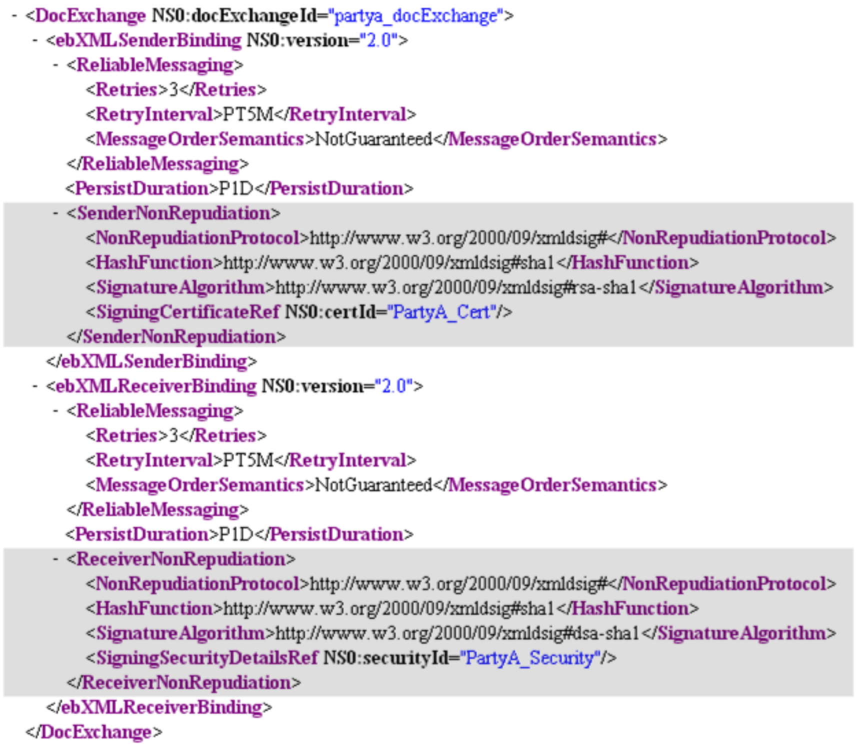


```
- <tp:ReceiverDigitalEnvelope>
    <tp:DigitalEnvelopeProtocol tp:version="1.0">S/MIME</tp:DigitalEnvelopeProtocol>
    <tp:EncryptionAlgorithm tp:minimumStrength="168" tp:oid="1.2.840.113549.3.7"
    tp:w3c="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"
    tp:enumerationType="encenum">DES-EDE3-CBC</tp:EncryptionAlgorithm>
    <tp:EncryptionCertificateRef tp:certId="Cyclone4_Cert"/>
  </tp:ReceiverDigitalEnvelope>
```

Figure 155. ReceiverDigitalEnvelope with S/MIME

Digital signature

The shaded areas of Figure 156 show digital signature.



```
- <DocExchange NS0:docExchangeId="partya_docExchange">
  - <ebXMLSenderBinding NS0:version="2.0">
    - <ReliableMessaging>
        <Retries>3</Retries>
        <RetryInterval>PT5M</RetryInterval>
        <MessageOrderSemantics>NotGuaranteed</MessageOrderSemantics>
      </ReliableMessaging>
      <PersistDuration>P1D</PersistDuration>
    - <SenderNonRepudiation>
        <NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</NonRepudiationProtocol>
        <HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</HashFunction>
        <SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#rsa-sha1</SignatureAlgorithm>
        <SigningCertificateRef NS0:certId="PartyA_Cert"/>
      </SenderNonRepudiation>
    </ebXMLSenderBinding>
  - <ebXMLReceiverBinding NS0:version="2.0">
    - <ReliableMessaging>
        <Retries>3</Retries>
        <RetryInterval>PT5M</RetryInterval>
        <MessageOrderSemantics>NotGuaranteed</MessageOrderSemantics>
      </ReliableMessaging>
      <PersistDuration>P1D</PersistDuration>
    - <ReceiverNonRepudiation>
        <NonRepudiationProtocol>http://www.w3.org/2000/09/xmldsig#</NonRepudiationProtocol>
        <HashFunction>http://www.w3.org/2000/09/xmldsig#sha1</HashFunction>
        <SignatureAlgorithm>http://www.w3.org/2000/09/xmldsig#dsa-sha1</SignatureAlgorithm>
        <SigningSecurityDetailsRef NS0:securityId="PartyA_Security"/>
      </ReceiverNonRepudiation>
    </ebXMLReceiverBinding>
  </DocExchange>
```

Figure 156. Digital signature

Managing CPAs

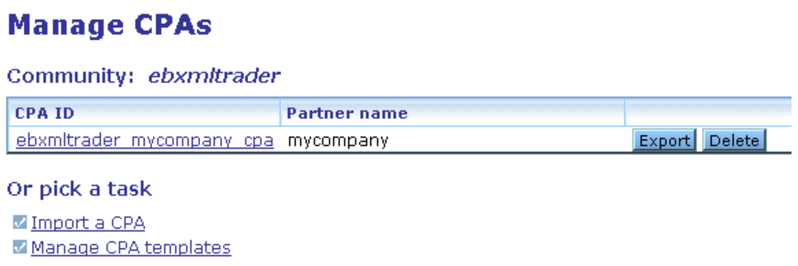
The **Manage CPAs** link at the bottom of the community summary page opens a page for viewing details of CPAs that have been imported and for importing CPAs and CPA templates.

You can view details of an imported CPA by clicking the CPA name.

The **Import a CPA** link is for importing a fully configured CPA that contains the trading details for your community and a partner. The **Manage CPA templates** link is for importing templates that partners can use with the partner registration wizard to build complete CPAs. See Self-registration of ebXML partners on page 483 for more information.

CPAs also can be imported automatically by copying the CPA files to [install directory]\[build number]\profiles\autoimport.

You also can add and remove CPAs using an API web service rather than through the user interface.


Manage CPAs

Community: *ebxmltrader*

CPA ID	Partner name	
ebxmltrader_mycompany_cpa	mycompany	Export Delete

Or pick a task

- Import a CPA
- Manage CPA templates

Figure 157. Manage CPAs page

Importing a CPA

Before you can import a CPA, it must be properly configured for your community and one partner. Building a CPA requires knowledge of the Collaboration-Protocol Profile and Agreement Specification Version 2.0 and the trading preferences of both parties.

Once a CPA has been created, you can import it to the trading engine and associate it with a community. Settings such as community name, routing ID and the delivery exchange for receiving messages must be the same in the community profile and CPA. The action of importing a properly configured CPA creates a partner profile for the partner specified in the CPA.

To import a CPA, click **Manage CPAs** at the bottom of the community summary page and click **Import a CPA**. Select Import the CPA from a file and type the path of the file to import or use the **Browse** button. Click **Import** to import the CPA.

Once imported, you can export a CPA to an XML file by clicking **Export**. You also can delete a CPA. If you want to change a CPA, delete it, change it and import it again. Removing a CPA deletes the CPA XML document, but does not delete the profile of the partner in the CPA.

Importing a CPA template

Before you can import a CPA template, it must be properly configured. Building a CPA template requires knowledge of ebXML standards. The reason for importing templates is so partners can use a registration wizard to enter their trading information. The system then completes the CPA, using the information the partner provided and information extracted from your community profile.

To import a CPA template, click **Manage CPAs** at the bottom of the community summary page and click **Manage CPA templates**. Type the path of the file to import or use the **Browse** button. Also type a description of the template. This description is how partners will recognize this template in the registration wizard. Click **Add** to import the template.

Once imported, you can delete a CPA template. If you need to change a template, delete the template, change it and import it again.

Web service API for CPAs

The ebXML CPA web service API allows you to add and remove CPAs through an application program interface instead of manually through the graphical user interface. This is useful if CPAs are generated by some management application and then pushed into the trading engine for activation.

The web service is described in CPAAPI.wsdl in [install directory]\[build number]\conf. There are two operations: **importCPA** and **removeCPA**.

You must write a web service client to use the API. Tools provide the functionality, through the WSDL2Java, to generate clients based on the WSDL file. One such tool is the open source AXIS tool (http://ws.apache.org/axis).

The trading engine server listens on the same port as the user interface. The default port is 6080. Make sure this port is accessible to the application that implements the client side of the web service.

HTTP is the transport protocol.

Any valid user of the trading engine can call the importCPA or removeCPA operation.

The importCPA and removeCPA operations do not return a result, except for a SOAP fault (CPAAPIInvocationException) in case of failure. The fault includes a string, code and details to identify the failure reason.

importCPA operation

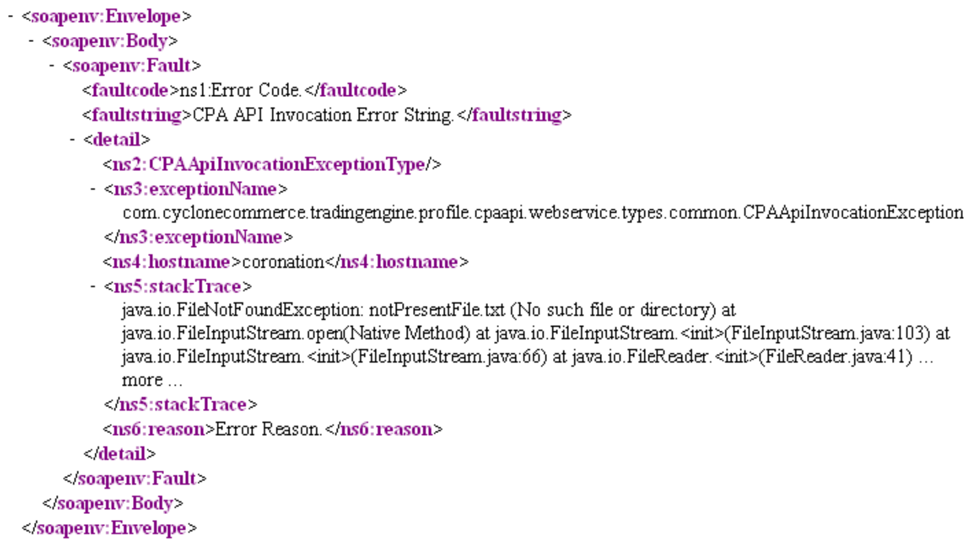
The import CPA operation has three parameters:

1. User name of type string
2. Password of type string
3. CPA of base 64 encoded string

The user name and password are required for authentication.

The CPA parameter is the actual CPA, encoded as a base 64 string. This is the CPA to be imported into the trading engine.

The return type is empty. In case of a failure, a WSDL fault is thrown back to the web service client. Figure 158 is an example of a SOAP fault.



```
- <soapenv:Envelope>
  - <soapenv:Body>
    - <soapenv:Fault>
        <faultcode>ns1:Error Code.</faultcode>
        <faultstring>CPA API Invocation Error String.</faultstring>
      - <detail>
          <ns2:CPAApiInvocationExceptionType/>
        - <ns3:exceptionName>
            com.cyclonecommerce.tradingengine.profile.cpaapi.webservice.types.common.CPAApiInvocationException
          </ns3:exceptionName>
          <ns4:hostname>coronation</ns4:hostname>
        - <ns5:stackTrace>
            java.io.FileNotFoundException: notPresentFile.txt (No such file or directory) at
            java.io.FileInputStream.open(Native Method) at java.io.FileInputStream.<init>(FileInputStream.java:103) at
            java.io.FileInputStream.<init>(FileInputStream.java:66) at java.io.FileReader.<init>(FileReader.java:41) ...
            more ...
          </ns5:stackTrace>
          <ns6:reason>Error Reason.</ns6:reason>
        </detail>
      </soapenv:Fault>
    </soapenv:Body>
  </soapenv:Envelope>
```

Figure 158. SOAP fault

The **faultstring** and **faultcode** elements of the SOAP fault indicate the error. If there is a reason and Java exception traceback, that is added to the **detail** element. The detail element has additional information, such as the exception name, which always is **CPAAPIInvocationException**.

Because the SOAP fault is caused by the web service, as opposed to a problem of the SOAP protocol, the Cyclone namespace is used. The namespace is: http://cyclonecommerce.com/tradingengine/profile/cpaapi/webservice.

For the importCPA operation, the following error strings (faultstring) and error codes (faultcode) are used.

Code	Error string	Description
100	Authentication failed	The server could not authenticate the user for the execution of the called operation.
200	CPA validation error	The CPA has not passed the CPA validation procedure. A traceback is provided for further information.

Code	Error string	Description
300	No matching local party found	The system could not locate a local party in the trading engine matching any party in the CPA. Either a community is missing or the CPA references wrong parties.
500	Unsupported Cyclone ebCPA feature	The ebXML CPA makes use of an unsupported ebCPA feature.
510	Unsupported Cyclone ebMS feature	The ebXML CPA requires ebMS features that are not supported.
520	Unsupported Cyclone ebBP feature	The ebXML CPA requires ebXML business process features that are not required yet.
530	Unsupported Cyclone ebRR feature	The ebXML CPA requires ebXML registry or repository support that is not supported.
600	Other CPA Import Error	A CPA-related error occurred while importing the CPA. A traceback is provided for further information.
601	Other System Error	A system-related error occurred while importing the CPA. A traceback is provided for further information.

removeCPA operation

The reomveCPA operation has three parameters:

1. User name of type string
2. Password of type string
3. CPA ID of type string

The user name and password are required for authentication.

The third argument is the CPA ID of the CPA to be removed from the trading engine. The CPA ID is the **cpaid** attribute of the **CollaborationProtocolAgreement** element of the CPA file. The data type of the CPA ID is string.

For the removeCPA operation, the following error strings (faultstring) and error codes (faultcode) are used.

Code	Error string	Description
100	Authentication failed	The server could not authenticate the user for the execution of the called operation.
200	CPA to be removed not present in system	The corresponding CPA of the CPA ID cannot be located in the trading engine.
600	Other CPA remove error	A CPA-related error occurred while removing the CPA. A traceback is provided for further information.
601	Other system error	A system-related error occurred while removing the CPA. A traceback is provided for further information.

STAR BODs with ebXML

Use this procedure to set up inline processing of outbound STAR BODs sent via the ebXML message protocol.

The trading engine can handle business object documents (BODs) that conform to Standards for Technology in Automotive Retail (STAR), the information technology standards body for the automotive industry.

The configuration is the same as for any community engaged in ebXML trading, with the additional step of setting up an inline processing action. This is so the trading engine can discern the sender, receiver, collaboration role and action.

This functionality provides XML schema validation and parses a STAR BOD for information needed to route an ebXML document. This makes it possible for a back-end system to provide only STAR BOD content, but not routing information.

This does not work as part of the ebXML message service handler. It is called before the MSH to discover information needed to process the message.

The trading engine identifies a STAR BOD in the following way:

1. The document must have a content type of **application/xml** or **text/xml**.

2 One of the following conditions must be met:

There is a meta-data element of **IsStarBOD** with a value of **true**, or

There is a namespace declaration in the XML equal to **http://www.starstandards.org/STAR**.

The following fields are parsed in an outbound STAR BOD:

ApplicationArea/Sender/DealerNumber

Sender from document location

ApplicationArea/Destination/DealerNumber

Receiver from document location

ApplicationArea/Sender/Task

Process specification from document location

DataArea/oa:*

Action from document location

Steps

1 On the community summary page, click **Message handler** in the navigation graphic at the top of the page.

2 Click **Add a message processing action** at the bottom of the page.

3 For the **Attribute** field, select **Content MIME type** and click **Next**.

4 On the operator and value page, the form should read “Process the message when Content MIME type Equals.” In the field after the word “Equals,” type **application/xml** and click **Next**.

5 Select **Perform inline processing via a Java class** and type the following in the **Class name** field:

com.cyclonecommerce.webservices.protocols.ebxml.inlineprocessing.StarBODProcessor

If you want the trading engine to validate the STAR BODs, type **validate** in the **Parameter** field. If you do, you must obtain the STAR BOD schemas and put them in your computer’s root directory (C:\ or /). Schemas are available from http://www.starstandard.org.

6 Click **Finish** to complete the action configuration.

HL7 payloads with ebXML

Use this procedure to set up inline processing for Health Level 7 version 2 and 3 payloads in conjunction with ebXML messaging. This functionality, which supports the HL7 Draft Standard for Trial Use, has been tested for interoperability. See http://www.drummondgroup.com.

The configuration is the same as for any community engaged in ebXML trading, with the additional steps of setting up an inline processing action and a CPA ID. This is so the trading engine can discern the CPA, sender, receiver, collaboration role and action.

Steps

1 On the community summary page, click **Message handler** in the navigation graphic at the top of the page.

2 Click **Add a message processing action** at the bottom of the page.

3 For the **Attribute** field, select **Content MIME type** and click **Next**.

4 On the operator and value page, the form should read “Process the message when Content MIME type Equals.” In the field after the word “Equals,” type **application/xml** and click **Next**.

5 Select **Perform inline processing via a Java class** and type the following in the **Class name** field:

com.cyclonecommerce.webservices.protocols.ebxml.inline processing.Hl7Processor

Leave the **Parameter** field blank.

6 Click **Finish** to complete the action configuration.

7 Set the CPA ID one of the following ways:

On the maintenance page for the integration pickup exchange, select the Message attributes tab and add an attribute named **CPAId**. Give the attribute a value equal to the CPA ID of the CPA you are using.

Or, set the CPA ID using an inline processing action.

Self-registration of ebXML partners

The partner registration wizard provides a way for one community (a hub) to get many partners configured to trade messages with it. The registration wizard employs a CPA template to build a complete CPA for the hub and each partner. Both the hub and each partner use the CPA tailored for their trading relationship to engage in ebXML trading.

This topic is for partners who want to trade via the ebXML message protocol. For AS1 or AS2, see Self-registration of AS1, AS2 partners on page 159.

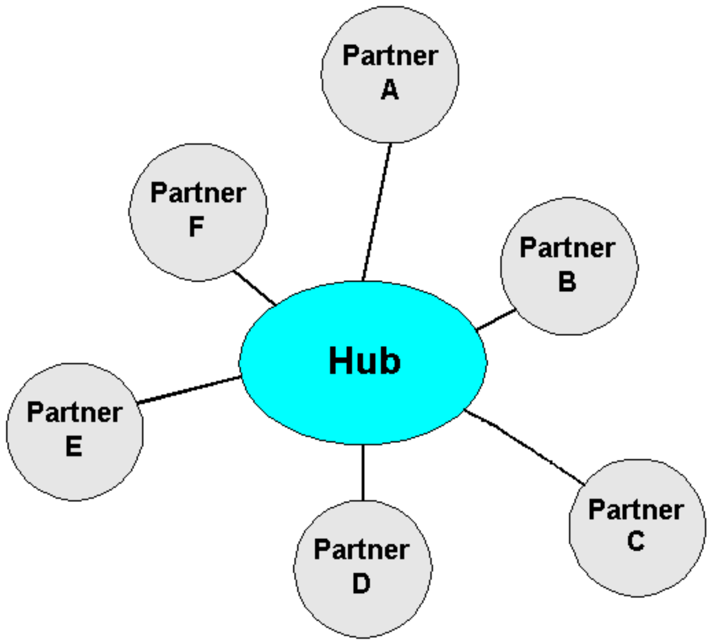


Figure 159. Hub-and-spoke trading network

The hub must take the first steps in setting up the ebXML hub-and-spoke network. Its trading engine must be installed and configured properly. The hub’s community profile must be configured.

The hub also must have at least one CPA template. Constructing a template requires a thorough understanding of ebXML practices. Such knowledge is a prerequisite for implementing Cyclone Activator for trading via ebXML.

Although spoke partners must be familiar with the operations of their trading engines, they do not need to know much about ebXML.

ebXML hub procedure

Use this procedure if you are an ebXML hub and want partners to use the registration wizard to build CPAs. These steps must be completed before partners can use the wizard.

Steps

1 Set a password for the **partner** user, if this has not already been done.

When you log on to the user interface for the first time after installing, there is a link on the getting started page for **Set a password for partner self-registration**. Click the link and type a password for the **partner** user. This link only appears if your user license allows you to run the partner registration wizard.

The system creates the partner user for you. Later, your partners will log on to your server's registration wizard with the user ID **partner** and the password you specify.

If the partner user already has been set up, check the users and roles area. Select **Users and roles** > **Manage users** or **Users and roles** > **partner registrant**.

2 Create and configure your community profile. This includes setting up delivery exchanges and a public-private key certificate for secure trading. The user interface provides guidance for creating and configuring a community. The user documentation also provides information. See The community profile on page 142.

Make sure the profile is fully configured. When partners use the registration wizard, information is extracted from your profile and combined with each partner's information to turn a CPA template into a completed CPA.

3 Import a CPA template document to your community. Go to the community summary page, click **Manage CPAs** at the bottom of the page, click **Mange CPA templates** and complete the fields for adding a template.

4 Give your partners the following information:

URL

The URL for connecting to the page for logging on to the registration wizard. The URL is in the following format:

http://host:6080/ui/

The variable **host** is the fully qualified domain name or IP address of the computer running the trading engine.

User name and password

The user name and password the partner must use for logging on to the registration wizard. Have the partner use **partner** and the password you specified for the partner user.

Community name

The name of the community the partner should select to join in the registration wizard.

Template name

The name of the CPA template the partner should select when using the registration wizard.

When a partner uses the registration wizard, the system uses the CPA template to build a complete CPA. Your system imports the CPA and creates a partner profile for the just-registered partner. Meanwhile, the wizard prompts the partner to save the CPA on the partner's local file system. If the partner uses Cyclone Interchange or Activator 5.0 or later and imports the CPA, the partner's system creates your profile based on the information in the CPA.

5. After a partner registers via the wizard, a message displays on the user interface home page, prompting you to approve the registration and associate the partner with your community. Click **Trading Partners** in the navigation graphic at the top of the community summary page, click **Add a partner to this community**, select **Choose an existing partner profile** and click **Next**. Select the partner and click **Add**.

ebXML partner procedure

Use this procedure if you are a spoke partner and want to use the hub's registration wizard to build a CPA that you can use to engage in ebXML trading with the hub partner.

1. If you use Cyclone Activator 5 or later, install the trading engine (see Installation on page 11) and log on to the user interface (see Open the user interface on page 22).

2. If you use Cyclone Activator 5 or later, create and configure your community profile. This includes setting up delivery exchanges and a public-private key certificate for secure trading. The user interface provides guidance for creating and configuring a community. The user documentation also provides information. See The community profile on page 142.

3. Export your encryption certificate and public key to a file. Include all certificates in the certification path, if possible.

   If you use Cyclone Activator 5 or later, export your community default encryption certificate and public key to a file. On the community summary page, click **Certificates** in the navigation graphic at the top of the page, click the certificate name and click **Export this certificate**. If you are exporting a self-signed certificate, you can export to a CER or PKCS #7 file. If you are exporting a third-party certificate, export to a PKCS #7 file and include all certificates in the certification path if possible.

   Keep this file. You will need it later when you use the registration wizard.

4. Collect and record the following information. This is information you need when using the registration wizard.

   - The name of the community the hub wants you to join. The hub must provide this information.
   - The name of the CPA template to use. The hub must provide this information.
   - Your community profile name.
   - Your community profile routing ID.
   - Your community profile contact name, phone number and e-mail address.
   - The URL the hub should use to send messages to you. This is a URL or e-mail address for the ebXML message protocol HTTP or SMTP transport. If you use Cyclone Activator 5 or later, you can find this by clicking **Delivery exchange** in the navigation graphic at the top of the community summary page. The URL or e-mail address is in the location column. You may want to consult with the hub about the URL to use.

5. Open a new browser session and connect to the URL the hub provided for the registration wizard.

6 To log on, type **partner** and the password the hub provided.

7 Follow the prompts to complete the registration wizard. When prompted to provide the URL for sending messages to your community, this can be a usual HTTP URL (for example, http://host.com:4080/exchange/1234) or for SMTP an e-mail address expressed as a URL (for example, mailto:company@mailserver.com).

8 When prompted, save the CPA to your file system.

9 Use the CPA to configure your trading engine to exchange ebXML messages with the hub.

If you use Cyclone Activator 5 or later, import the CPA. Go to the community summary page, click **Manage CPAs** at the bottom of the page, click **Import a CPA** and complete the field for importing a CPA from a file. If the CPA is successfully imported, the system generates a partner profile for the hub and associates the profile with your community.

ebXML troubleshooting

The first thing you can do for ebXML troubleshooting is turn on debug level event messaging. This results in verbose messages about ebXML activity being written to system log files. Do the following to enable debug level ebXML events.

1 Open for editing:

[install directory]\[build number]\conf\log4j.properties

2 Under the section of the file titled “you can change levels of categories below this line,” find the following line:

log4j.category.com.cyclonecommerce.webservices=info

3 Change the **info** value to **debug**.

4 Save and close the file.

The following are some common ebXML issues and possible solutions.

No binary collaboration found message

Add an ebXML delivery exchange

Receiver routing ID is missing message

The receiver can be your community or a partner. This message indicates the routing ID for one or the other is unknown to the trading engine. This could be because a routing ID for one or the other has not been defined.

CPA not found for CPA ID message

Make sure the CPA ID is specified in the meta-data (MMD, JMS property or message attribute for the delivery exchange).

CanSend cannot be found for action message

CanSend Action is not defined in the CPA for the community.

CPA-related issues

Trading may fail because of errors or omissions in a CPA. Check a CPA thoroughly to make sure the document is accurate and complete.

❖ ❖ ❖

28 RosettaNet support

This chapter describes how to use the trading engine to exchange documents via the RosettaNet message protocol. Your organization must have a thorough understanding and working knowledge of RosettaNet to successfully trade documents using this protocol. For information about RosettaNet see www.rosettanet.org.

Note: RosettaNet requires a special user license to enable the protocol in the trading engine.

Concepts

RosettaNet overview

RosettaNet is the name both of a consortium of companies and of a set of processes and standards for conducting electronic business transactions. Cyclone Activator supports two versions of the RosettaNet Implementation Framework (RNIF). RNIF 1.1 and 2.0 are implementation guidelines for companies that want to create interoperable software that execute Partner Interface Processes (PIPs).

Cyclone Activator is the packaging and transport interface to the back-end PIP engine. The back-end determines what message is next in the PIP process, whether inbound or outbound. The back-end also can optionally generate Message Meta-data Documents and submit them to Cyclone Interchange. MMDs are interface XML documents that bind the RosettaNet headers (preamble, service header, delivery header) and the service content.

There are two broad categories of messages involved in the exchange of PIP business documents. They are business action messages and business signal messages.

Business actions are messages with a business-related content, such as a purchase order or request for quote. DTDs and message guidelines for their corresponding PIPs define these messages.

Business signals are positive or negative acknowledgments sent in response to a business action. Signal messages, which are part of RNIF, are never acknowledged.

A request is the act of initiating some aspect of a business process. A response is the act of responding to a request. Business action messages are used to implement requests and responses. Figure 160 shows a message flow.

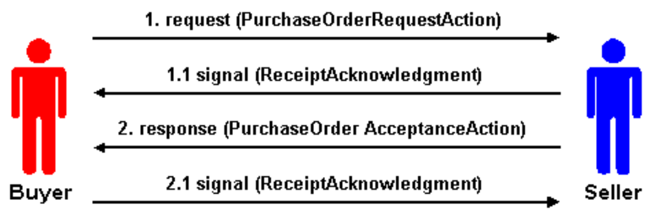

1. request (PurchaseOrderRequestAction)
1.1 signal (ReceiptAcknowledgment)
2. response (PurchaseOrder AcceptanceAction)
2.1 signal (ReceiptAcknowledgment)
Buyer
Seller

Figure 160. RosettaNet messages

Cyclone Activator does duplicate checking for RosettaNet. The unique identification for a RosettaNet message is the core ID. This cannot be disabled in Cyclone Activator. RosettaNet requires that the PIP instance ID be unique across all PIP instances and that the message ID be unique within that PIP.

RosettaNet configuration outline

The following outlines the steps for configuring a community to trade via the RosettaNet protocol.

1. Install the trading engine. See Installation on page 11.

2. Log on to the user interface. See Open the user interface on page 22.

3. Create and configure your community profile. This includes setting up delivery exchanges and a public-private key certificate for secure trading. The user interface provides guidance for creating and configuring a community. The user documentation also provides information. See The community profile on page 142.

   For the delivery exchange for receiving messages from partners, select RosettaNet 1.1 or RosettaNet 2.0 as the message protocol. See Trading delivery exchanges on page 183.

If you need to produce message meta-data documents (MMDs) when routing messages from partners to a back-end file system, select “file system with message metadata” as the inbound integration transport. If you do not need MMDs, you can use any other inbound integration transport. See Integration exchanges on page 185.

If a partner uses Cyclone Interchange or Activator 4.x, be aware of the following. If the community profile you export has delivery exchanges for both RosettaNet 1.1 and 2.0 for receiving messages, the partner’s trading engine will ignore one of the exchanges upon importing the profile. Consult with your partner about which exchange to use, and inform the partner to edit the partner profile accordingly.

4. Set up partner profiles by importing profile files or manually adding them. Associate the partners with your community. See Add a partner on page 150.

   Make sure the message protocol for sending messages to partners is RosettaNet 1.1 or RosettaNet 2.0.

   If a partner uses Cyclone Interchange or Activator 4.x, the partner profile you import will have delivery exchanges for sending messages for both RosettaNet 1.1 and 2.0. Consult with your partner about which exchange to use, and delete the unused exchange in the profile.

5. Consult with your partners on the PIPs to use.

6. Add PIP DTDs to the trading engine dtds directory. See Add a PIP on page 491.

7. Check collaboration settings for document encrypting and signing options of outbound documents. You can configure settings by partner rather than use default settings. See RosettaNet 1.1 default settings on page 388 or RosettaNet 2.0 default settings on page 389.

8. Define the PIPs in the pipdefinitions.xml file. See Configuring the pipdefinitions.xml file on page 492.

   The community now is ready to begin trading messages.

Add a PIP

Use this procedure to use a PIP not already supported by Cyclone Activator.

The trading engine supports a number of PIPs. The PIP document type definitions (DTDs) are in [install directory]\common\conf\rosettanet\dtds.

Steps

1 Obtain the correct version of the PIP package from RosettaNet at www.rosettanet.org.

A PIP package is a zip file containing the PIP specification in a Word document, one or more XML DTDs and one or more HTML message guidelines files. The DTDs, one per PIP message, define the structure of the messages involved in the PIP. The HTML message guidelines files, one per PIP message, describe the elements of the message in detail.

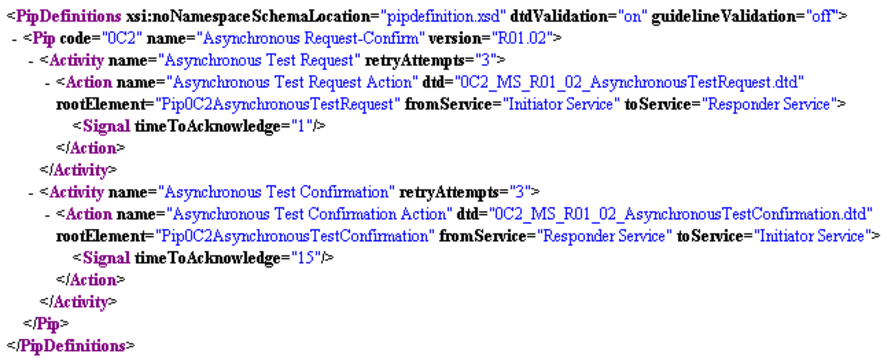
2 Copy the DTDs to [install directory]\common\conf\rosettanet\dtds.

3 Define the PIP in the pipdefinitions.xml file. See Configuring the pipdefinitions.xml file on page 492.

Configuring the pipdefinitions.xml file

The **pipdefinitions.xml** file is the key for configuring the trading engine for RosettaNet trading, although other steps are required as well. The file defines the business messages to be traded for each PIP and the order in which each message is sent or received. The trading engine uses the definitions to validate RosettaNet messages and the trading order.

Figure 161 shows an example of the pipdefinitions.xml file. This example shows the definition for using the Asynchronous Request-Confirm test PIP. The PIP is defined between the beginning and ending Pip element tags. You can define as many PIPs as you want in this file.



```
<PipDefinitions xsi:noNamespaceSchemaLocation="pipdefinition.xsd" dtdValidation="on" guidelineValidation="off">
 - <Pip code="0C2" name="Asynchronous Request-Confirm" version="R01.02">
   - <Activity name="Asynchronous Test Request" retryAttempts="3">
     - <Action name="Asynchronous Test Request Action" dtd="0C2_MS_R01_02_AsynchronousTestRequest.dtd"
       rootElement="Pip0C2AsynchronousTestRequest" fromService="Initiator Service" toService="Responder Service">
         <Signal timeToAcknowledge="1"/>
       </Action>
     </Activity>
   - <Activity name="Asynchronous Test Confirmation" retryAttempts="3">
     - <Action name="Asynchronous Test Confirmation Action" dtd="0C2_MS_R01_02_AsynchronousTestConfirmation.dtd"
       rootElement="Pip0C2AsynchronousTestConfirmation" fromService="Responder Service" toService="Initiator Service">
         <Signal timeToAcknowledge="15"/>
       </Action>
     </Activity>
   </Pip>
</PipDefinitions>
```

Figure 161. Example of pipdefinitions.xml file

The following describes the elements in the pipdefinitions.xml file. See Figure 161 for an example of how the elements are used. Most of the information is used to build RosettaNet headers. But some elements, such as retryAttempts and timeToAcknowledge, affect performance of the trading engine.

PipDefinitions

The PIP definitions are placed between the beginning and ending PipDefinitions elements.

xsi:noNamespaceSchemaLocation

The default schema is **pipdefinition.xsd**. The file is at [install directory]\[build number]\conf\rosettanet\schemas.

dtdValidation

Indicates (**on** or **off**) whether the trading engine validates the document against a DTD. If **on**, copy the DTD to [install directory]\common\conf\rosettanet\dtds.

guidelineValidation

Indicates (**on** or **off**) whether the trading engine validates the document against a guidelines xml file. If **on**, copy the file to [install directory]\common\conf\rosettanet\guidelines.

Pip

Each PIP is defined between beginning and ending Pip elements.

code

An attribute of the Pip element, this code identifies the PIP. The code is available in the RosettaNet PIP specification document. For example, for the Asynchronous Request-Confirm PIP, the code is **0C2**.

This is required to build the RosettaNet headers. It is not specified by the service content. The trading engine determines the PIP code and PIP version based on the document type of the service content.

name

An attribute of the Pip element, this is the name of the PIP. The name is available in the RosettaNet PIP specification document.

version

An attribute of the Pip element, this is the version of the PIP. The version is available in the RosettaNet PIP specification document. For example, for the Asynchronous Request-Confirm PIP, the version is **R01.02**.

This is required to build the RosettaNet headers. It is not specified by the service content. You can have multiple versions of a PIP, but the document types need to be different. RosettaNet usually gives service contents a different document type based on the version.

Activity

The activity as defined in the RosettaNet PIP specification document. A PIP can have multiple activities. Each activity must be listed in the pipdefinitions.xml file in parallel with the specification.

name

An attribute of the Activity element, this is the activity name as defined in the RosettaNet PIP specification document.

retryAttempts

An attribute of the Activity element, this specifies how many times to resend a message when a signal is not received within the time specified in the timeToAcknowledge attribute. This value is specified in the RosettaNet PIP specification document.

Action

The action for the parallel activity as specified in the RosettaNet PIP specification document

name

An attribute of the Action element, this is the name of the action. This is the business message payload. The name is available in the RosettaNet PIP specification document.

dtd

This is an attribute of the Action element. If dtdValidation="on", the value is the name of the dtd to validate against. The name is available in the RosettaNet PIP specification document. The dtd must be copied to [install directory]\[build number]\conf\rosettanet\dtds.

If validation is off, this attribute is not used.

rootElement

An attribute of the Action element, the rootElement is used to identify the action when there is no DOCTYPE definition. The value of rootElement is in the service content.

This handles the case where there is no DOCTYPE definition and provides for a means to validate the XML (given the dtd attribute). If there is a DOCTYPE definition, the DOCTYPE is used to validate the XML, otherwise the dtd attribute will be used. The document is rejected if there is no DOCTYPE, DTD validation is on and the dtd attribute is missing or blank.

docType

An attribute of the Action element, this is the name of the dtd for the action. The name is available in the RosettaNet PIP specification document. The dtd must be copied to [install directory]\[build number]\conf\rosettanet\dtds.

fromService

An attribute of the Action element, this is the service from which the message is being sent. This value is specified in the RosettaNet PIP specification document.

This is required to build the RosettaNet headers. It is not specified by the service content.

toService

An attribute of the Action element, this is the service to which the message is being sent. This value is specified in the RosettaNet PIP specification document.

This is required to build the RosettaNet headers. It is not specified by the service content.

Signal

This element specifies the conditions for the response to the message action.

timeToAcknowledge

An attribute of the Signal element, this specifies the time in minutes within which the message acknowledgment must be received. If not received within the specified time, the message is to be resent up to the limit in the retryAttempts attribute. After each resend, the timeToAcknowledge interval must elapse before another resend is attempted. This value is specified in the RosettaNet PIP specification document.

RNIF meta-data elements

The following are the available RNIF meta-data elements.

These elements are listed in the correct format. When using meta-data elements, make sure to use the proper case.

Description of elements

The following are the meta-data elements. In the case of MMDs, some exceptions apply. See MMD elements on page 500.

BusinessActivityIdentifier

RosettaNet activity identifier of the message as defined in the PIP specification. Required.

BusinessProtocolVersion

The version of the RNIF messages to be traded, either **1.1** or **2.0**. Required.

FromGlobalPartnerClassificationCode

The role specified in the PIP. This value is from the service header.

FromGlobalSupplyChainCode

A value from the service content.

FromRole

The role the trading partner sending the message plays in this PIP. Optional.

Overrides GlobalPartnerRoleClassificationCode in the service content in this hierarchy:

```
fromRole
        GlobalPartnerRoleClassificationCode
```

FromService

The service from which the message is being sent, as defined in the PIP specification. Required.

GlobalBusinessActionCode

The action code corresponding to the action to which the message is in reply, as defined in the PIP specification. Required.

GlobalBusinessSignalCode

The signal code, if the message is a signal. This is parsed from inbound signal messages. It is not specified in an MMD.

GlobalDocumentFunctionCode

Required for RNIF 1.1 action messages.

GlobalUsageCode

Determines whether the message is to be used in **Test** mode or in **Production** mode. Required.

InitiatingRoutingId

Specifies the initiating routing ID.

InitiatingRoutingIdKnown

Specifies whether the initiating partner is known.

InReplyToActionCode

This code is parsed from a signal message. The code is specified in the PIP specification.

InReplyToTrackingId

Helps to identify the message to which this message is in reply.

PipCode

RosettaNet PIP Code of the message. Set by the initiating partner. Defines what PIP code is being traded. Required.

PipInstanceId

The ID of this PIP instance. This must be unique within the context of the initiating partner. Optional.

PipVersion

RosettaNet PIP Version of the message. Set by the initiator of this transaction. Required.

MessageTrackingId

Uniquely identifies the message for tracking purposes. Must be unique within the context of the message sender. This value is parsed from the delivery header.

OriginalMessageDigest

The MIC value of the original message (in the signal).

ReceiverRoutingId

The ID of the receiving partner. This must be a DUNS number. Optional.

If not specified, the value comes from the GlobalBusinessIdentifer in the service content in this hierarchy:

```
toRole
    PartnerRoleDescription
        PartnerDescription
            BusinessDescription
                GlobalBusinessIdentifier
```

SenderRoutingId

The ID of the sending partner. This must be a DUNS number. Optional.

If not specified, the value comes from the GlobalBusinessIdentifer in the service content in this hierarchy:

```
fromRole
    PartnerRoleDescription
        PartnerDescription
            BusinessDescription
                GlobalBusinessIdentifier
```

service-content

The mandatory payload ID of the service content in an MMD.

ServiceContentDocType

The document type extracted from the service content.

SignalDigestAlg

Defines the algorithm to use for signing a signal (the same as the one used for the original message).

SignalEncryptionAlg

Defines the algorithm to use for encrypting a signal (the same as the one used for the original message).

SignalEncryptionStrength

Defines the algorithm strength to use for encrypting a signal (the same as the one used for the original message).

ToGlobalPartnerClassificationCode

The role specified in the PIP. This value is from the service header.

ToGlobalSupplyChainCode

A value from the service content.

ToRole

The role the trading partner receiving the message plays in this PIP. Optional.

Overrides GlobalPartnerRoleClassificationCode in the service content in this hierarchy:

```
toRole
          GlobalPartnerRoleClassificationCode
```

ToService

The service to which the message is being sent, as defined in the PIP specification. Required.

TransactionIdentity

MMD elements

For MMDs, some names of meta-data elements are slightly different than those listed in Description of elements on page 496, but the usage is the same. The following lists these discrepancies.

ToRoutingId

Same as ReceiverRoutingId on page 498.

FromRoutingId

Same as SenderRoutingId on page 498.

ToClassificationCode

Same as ToGlobalPartnerClassificationCode on page 499.

FromClassificationCode

Same as FromGlobalPartnerClassificationCode on page 496.

KnownInitiatingRoutingId

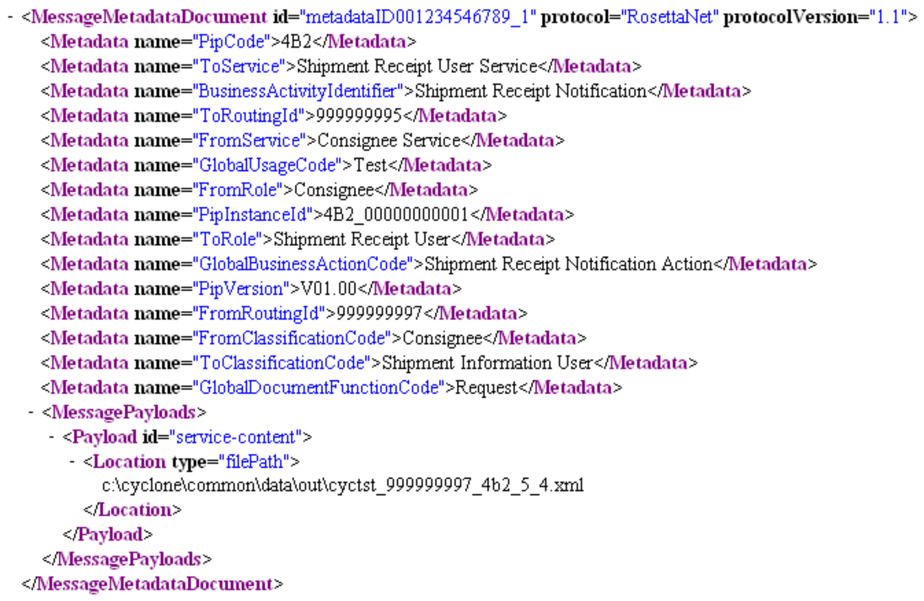
Same as InitiatingRoutingIdKnown on page 497.

Message meta-data documents

In file system integration with a back-end system, the trading engine supports RNIF using message meta-data documents (MMDs) as the interface between it and the back-end. The MMDs are XML documents that point to a RosettaNet document on a file system and contain information the trading engine uses to process documents.

The trading engine generates MMDs for the documents it sends to a back-end system. To do this you must use the file system with message meta-data integration option. Your back-end system must generate the MMDs for documents the trading engine retrieves from integration.

Figure 162 is an example of an MMD associated with a RosettaNet document. For meta-data descriptions, see RNIF meta-data elements on page 496.



```
- <MessageMetadataDocument id="metadataID001234546789_1" protocol="RosettaNet" protocolVersion="1.1">
    <Metadata name="PipCode">4B2</Metadata>
    <Metadata name="ToService">Shipment Receipt User Service</Metadata>
    <Metadata name="BusinessActivityIdentifier">Shipment Receipt Notification</Metadata>
    <Metadata name="ToRoutingId">999999995</Metadata>
    <Metadata name="FromService">Consignee Service</Metadata>
    <Metadata name="GlobalUsageCode">Test</Metadata>
    <Metadata name="FromRole">Consignee</Metadata>
    <Metadata name="PipInstanceId">4B2_00000000001</Metadata>
    <Metadata name="ToRole">Shipment Receipt User</Metadata>
    <Metadata name="GlobalBusinessActionCode">Shipment Receipt Notification Action</Metadata>
    <Metadata name="PipVersion">V01.00</Metadata>
    <Metadata name="FromRoutingId">999999997</Metadata>
    <Metadata name="FromClassificationCode">Consignee</Metadata>
    <Metadata name="ToClassificationCode">Shipment Information User</Metadata>
    <Metadata name="GlobalDocumentFunctionCode">Request</Metadata>
  - <MessagePayloads>
    - <Payload id="service-content">
      - <Location type="filePath">
          c:\cyclone\common\data\out\cyctst_999999997_4b2_5_4.xml
        </Location>
      </Payload>
    </MessagePayloads>
  </MessageMetadataDocument>
```

Figure 162. Example of an MMD for RosettaNet

You should use MMDs when you want to override data in service content or when there are payloads in addition to the service content. In the latter case, when it is not necessary to override service content data, you do not have to specify meta-data in the MMD. The MMD in such a case acts as the glue between the service content and additional payloads.

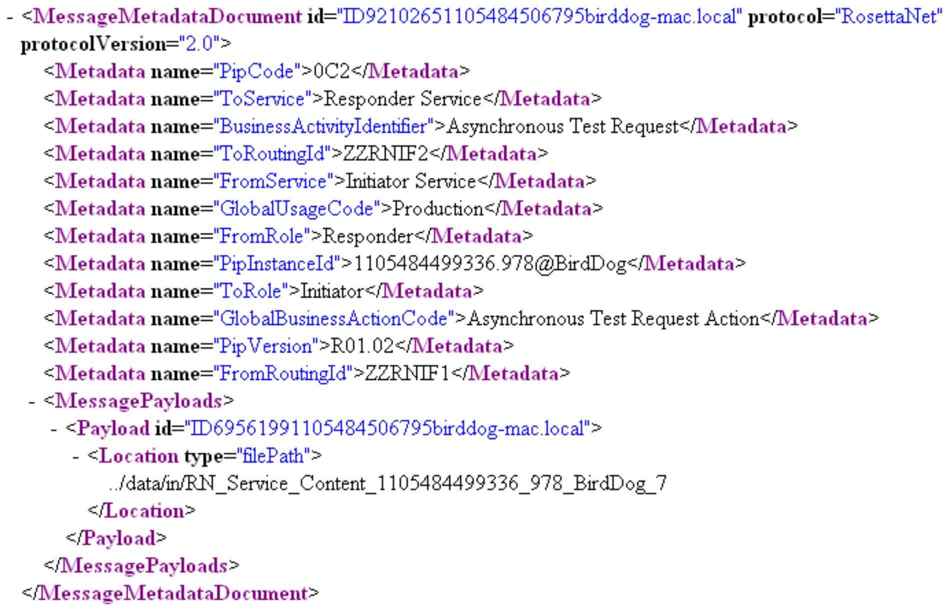
You do not have to use MMDs when the service contents contains all the data needed to generate headers and there are no payloads in addition to the service content.

Special handling of meta-data

Because of vagaries in the RNIF standard, Cyclone Activator has devised a way to pass the GlobalUsageCode and PipInstanceId to and from integration. There are two use cases: with MMDs and without MMDs. You need to know so your back-end system can deal with either case.

With MMDs

For inbound integration, if MMDs are used the trading engine produces an MMD to integration containing values for the GlobalUsageCode and PipInstanceId (Figure 163). For outbound integration the GlobalUsageCode and PipInstanceId can be included in the MMD. For response messages the PipInstanceId also must be included in the MMD. If included within the MMD, the GlobalUsageCode and PipInstanceId values from the MMD are used in the packaged RNIF message.



```
- <MessageMetadataDocument id="ID921026511105484506795birddog-mac.local" protocol="RosettaNet"
  protocolVersion="2.0">
    <Metadata name="PipCode">0C2</Metadata>
    <Metadata name="ToService">Responder Service</Metadata>
    <Metadata name="BusinessActivityIdentifier">Asynchronous Test Request</Metadata>
    <Metadata name="ToRoutingId">ZZRNIF2</Metadata>
    <Metadata name="FromService">Initiator Service</Metadata>
    <Metadata name="GlobalUsageCode">Production</Metadata>
    <Metadata name="FromRole">Responder</Metadata>
    <Metadata name="PipInstanceId">1105484499336.978@BirdDog</Metadata>
    <Metadata name="ToRole">Initiator</Metadata>
    <Metadata name="GlobalBusinessActionCode">Asynchronous Test Request Action</Metadata>
    <Metadata name="PipVersion">R01.02</Metadata>
    <Metadata name="FromRoutingId">ZZRNIF1</Metadata>
  - <MessagePayloads>
    - <Payload id="ID695619911105484506795birddog-mac.local">
      - <Location type="filePath">
          ../data/in/RN_Service_Content_1105484499336_978_BirdDog_7
        </Location>
      </Payload>
    </MessagePayloads>
  </MessageMetadataDocument>
```

Figure 163. MMD produced for inbound integration

Without MMDs

If MMDs are not used, the header of outbound messages must have the following information:

```
<thisDocumentIdentifier>
  <ProprietaryDocumentIdentifier>
    [proprietary document ID]:[optional GlobalUsageCode]:[optional
    PipInstanceId]
  </ProprietaryDocumentIdentifier>
</thisDocumentIdentifier>
```

The value of the ProprietaryDocumentIdentifier element should take the form of a colon-separated list of strings. The first string is the proprietary document ID. Optionally, the GlobalUsageCode can be included second. And, optionally, the PipInstanceId included third. The PipInstanceId can only be included when the GlobalUsageCode is also included.

The proprietary document identifier, GlobalUsageCode and PipInstanceId values from the outbound message are used in the packaged RNIF message.

For inbound messages, the value of the ProprietaryDocumentIdentifier element is unmodified from what was included by the sender of the message. If the proprietary document ID, GlobalUsageCode and PipInstanceId where included by the sender, the values are included in the message produced to integration.

❖ ❖ ❖

29 CSOS orders

For licensed users, the trading engine supports digital signing and verification of controlled substance orders in compliance with the Controlled Substance Ordering System of the U.S. Drug Enforcement Administration. The documents being handled are purchase orders that conform to CSOS standards. These can be EDI X12 or XML documents.

This mortar and pestle icon on the user interface toolbar indicates your user license supports CSOS functionality. If this icon is absent, you are not licensed for CSOS processing.

Concepts

Procedure

Overview of CSOS functionality

With digital certificates issued by DEA, a user can sign controlled substance orders before the trading engine sends the orders to partners. Orders are signed with the user's private key corresponding to the user's public-private key pair in the certificate. Once signed, the user’s certificate, containing the public key only, is transmitted with the order.

For users who receive signed controlled substance orders from partners, the trading engine validates the orders as authentic and unaltered.

When the trading engine validates the signature of received orders, it checks the signer's certificate against a DEA list to make sure the certificate has not been revoked. This is done using a certificate revocation list (CRL). The CRL is issued and updated by DEA. The trading engine has the ability to retrieve the CRL over the Internet.

After authenticating the received order, the trading engine makes a back up copy of the order and the public key and certificate. The default behavior of the trading engine is to back up received messages. But make sure backing up is enabled for the delivery exchange for receiving messages from partners to ensure CSOS compliance.

Before configuring the trading engine to handle CSOS orders, your organization must comply with DEA's rules for CSOS participants, including obtaining a digital signing certificate from DEA.

How it works

Figure 164 illustrates how CSOS functionality works in the trading engine according to the following scenario.

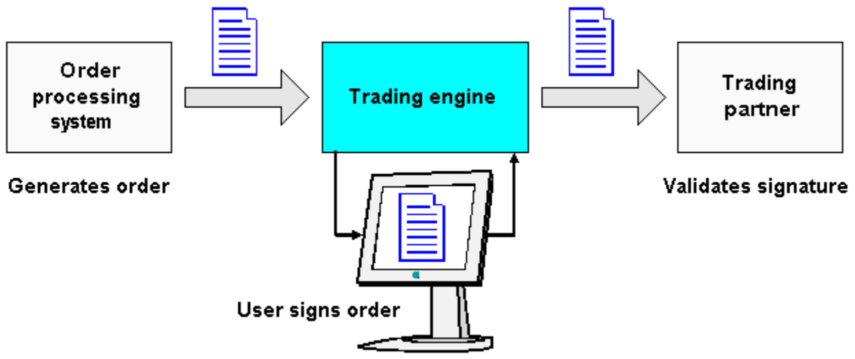


Figure 164. Flow of CSOS orders

1 A back-end system generates a purchase order that conforms to CSOS standards.

2 The trading engine picks up the document. Recognizing it as a CSOS order, the trading engine places the document in a signing queue. This action suspends the usual processing routine of packaging documents for sending to partners.

Only the documents you want are handled in this manner. Other documents are not affected.

3 A CSOS user logs on to the trading engine user interface to check for CSOS orders awaiting approval.

4 The user displays a pending order in the user interface. The system renders the document in an easy to read format, provided a stylesheet has been applied to the document type.

5. The user types a password and approves the order. This action results in the trading engine using the user's private key to digitally sign the order.

6. The signed order is released for the trading engine to continue outbound processing.

7. The partner's system, upon receiving the signed order, verifies the signature using DEA-issued root and intermediate certificates. The receiver also checks whether the DEA registration number is the same in the order and the root certificate. Orders that fail verification are rejected.

CSOS configuration for sending

Follow this outline to configure the trading engine to sign and send CSOS purchase orders.

1. Install the trading engine and set up community and partner profiles in the usual manner.

   See the following topics:

   System requirements on page 1
   Installation on page 11
   Getting started on page 133

2. Obtain a user signing certificate from DEA. A signing certificate is needed to send signed orders to a partner.

3. Import the signing certificate to the trading engine. See Import CSOS signing certificate on page 506.

   The required DEA-issued intermediate and root certificates are pre-loaded in the trading engine. These are required to complete the chain of trust upon importing the DEA-issued signing certificate (also known as an end-entity certificate).

4. Configure the trading engine to identify outbound CSOS orders and place the documents in a queue for signing. See Identify CSOS purchase orders on page 507.

5. Display and approve the orders in the user interface. See Sign pending orders on page 519.

CSOS configuration for receiving

Follow this outline to configure the trading engine to receive signed CSOS purchase orders from partners.

1. Install the trading engine and set up community and partner profiles in the usual manner.

   See the following topics:

   System requirements on page 1
   Installation on page 11
   Getting started on page 133

   The required DEA-issued intermediate and root certificates are pre-loaded in the trading engine. These are required to verify the authenticity of the inbound signed CSOS documents.

2. Verify CRL checking is active. See CSOS certificate revocation lists on page 507.

3. Configure the trading engine to identify CSOS orders received from partners. See Identify CSOS purchase orders on page 507.

Import CSOS signing certificate

Use this procedure to import the DEA-issued signing certificate to the trading engine. This certificate is assigned to a user who uses it to sign CSOS orders before the trading engine sends the orders to partners.

CAUTION: The pfx or p12 file that DEA provides contains your private encryption key. Make sure to securely protect this file. Do not share it with anyone.

Steps

1. Select **Users and Roles** > **Manage users** on the toolbar.

2. Click the name of the user to whom you will assign the certificate.

3. Click **Add a certificate**.

4. Select **Import a certificate and private key from a file** and click **Next**.

5. Type the path to the certificate file or use the **Browse** button to locate the file. Type the password. Click **Next**.

   A name for the certificate is displayed. You can use this name or type a different name.

6. Click **Finish** to import the certificate.

   To verify the certificate has been imported, click the **Personal certificates** tab and look for the name of the certificate you just imported.

CSOS certificate revocation lists

DEA maintains certificate revocation lists of CSOS certificates it has issued that have been revoked for one reason or another. Revoked certificates are not valid for CSOS order signing. The trading engine fails a CSOS order signed with a revoked user certificate.

Whether you are sending or receiving signed orders, you need to have CRL checking in place. With CSOS functionality enabled in your user license, CRL checking is active by default. The trading engine reads a URL in the CSOS user certificate and downloads a CRL to use in checking for invalid certificates. CRL files are stored in [install directory]\common\conf\crls.

To verify CRL checking is active, open the crossworks.properties file in [install directory]\[build number]\conf. Make sure that crls.require=true and crls.autoRetrieve=true.

See Certificate revocation lists on page 359 for more details about CRLs.

Identify CSOS purchase orders

The identify CSOS purchase orders page in the user interface is used to:

- Identify EDI X12 850 documents or XML purchase order documents for handling as CSOS documents.
- Assign a document type to CSOS documents. This associates documents to stylesheets that format XML files for viewing in the user interface. An EDI document is transformed dynamically to XML before a stylesheet is applied.
- Identify the delivery exchange where the trading engine obtains CSOS documents. The sending and receiving parties also can be specified.

- If you receive signed orders, specify whether to accept or reject duplicate CSOS messages.

The identify CSOS purchase orders page has three tabs:

- Document types tab
- Order sources tab
- CSOS duplicate orders tab

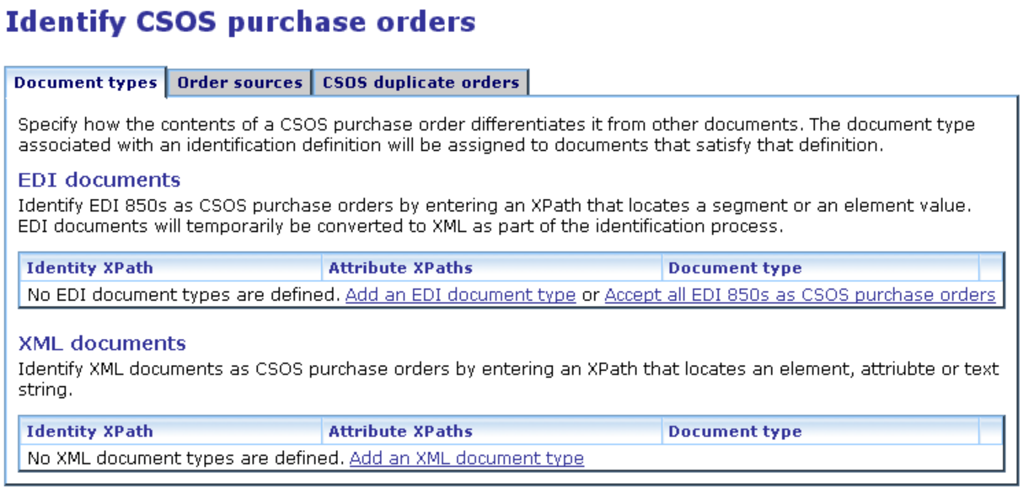
If you send or receive signed orders, you must use the document types tab to identify CSOS documents. The order sources tab allows you to set additional filtering conditions. Whether you should use the order sources tab depends on your situation. For instance, if you receive CSOS orders via certain message protocols — AS1, AS2, Secure file, Secure e-mail, ebXML — a content MIME type of application/x-csos-signed-order is included in inbound message headers. This triggers the trading engine to validate and unpack the inbound messages as CSOS documents, and using the order sources tab is unnecessary. On the other hand, if you send CSOS orders, you can use the order sources tab to specify a particular integration exchange for picking up documents identified on the document types tab.

If you receive signed orders from partners, use of the CSOS duplicate orders tab is optional. If you only send signed orders, ignore this tab.

The following topics describe each tab of the identify CSOS purchase orders page.

Document types tab

The document types tab of the identify CSOS purchase orders page (Figure 165) lets you specify EDI X12 850 documents or XML purchase order documents as valid CSOS documents. The tab is divided into sections for separately identifying EDI and XML documents. You need to use this tab if you send or receive signed orders.


Identify CSOS purchase orders

Document types | Order sources | CSOS duplicate orders

Specify how the contents of a CSOS purchase order differentiates it from other documents. The document type associated with an identification definition will be assigned to documents that satisfy that definition.

EDI documents

Identify EDI 850s as CSOS purchase orders by entering an XPath that locates a segment or an element value. EDI documents will temporarily be converted to XML as part of the identification process.

Identity XPath	Attribute XPaths	Document type
No EDI document types are defined. Add an EDI document type or Accept all EDI 850s as CSOS purchase orders		

XML documents

Identify XML documents as CSOS purchase orders by entering an XPath that locates an element, attriubte or text string.

Identity XPath	Attribute XPaths	Document type
No XML document types are defined. Add an XML document type		

Figure 165. Document types tab of Identify CSOS purchase orders page

Using the document types tab requires some familiarity with XML Path Language (XPath). XPath uses a file's logical hierarchy to find elements in XML documents. The user interface has an XPath wizard to help you add XPath strings using your sample documents. The wizard temporarily transforms EDI documents to XML to let you add XPaths. When the trading engine performs CSOS processing, it also transforms EDI to XML, using the XPaths to parse data in a document. XML documents, of course, already are XML and do not require such transformation.

Converting EDI to XML for the purpose of assigning XPaths is preferred for its reliability over parsing of raw EDI documents.

For more information about XPath, see http://www.w3.org/TR/xpath.

The following topics describe how to use the tab for identifying EDI or XML documents for CSOS processing.

Identifying EDI documents

There are two ways to identify 850 documents for CSOS handling.

1. Specify that only 850s that meet certain conditions are handled as CSOS documents. 850s not meeting the conditions are passed over.

   To use this option, click **Add an EDI document type**. This displays a section (Figure 166) that has a field for an identity XPath that can be used to differentiate one type of document from another. There also are fields for specifying XPaths for the DEA registration number and order number and for selecting a document type, which is used to format a document for viewing in the user interface.

You can add multiple EDI document definitions if you select an identity XPath for each one. The identity XPath provides a way to identify different varieties of 850 documents.

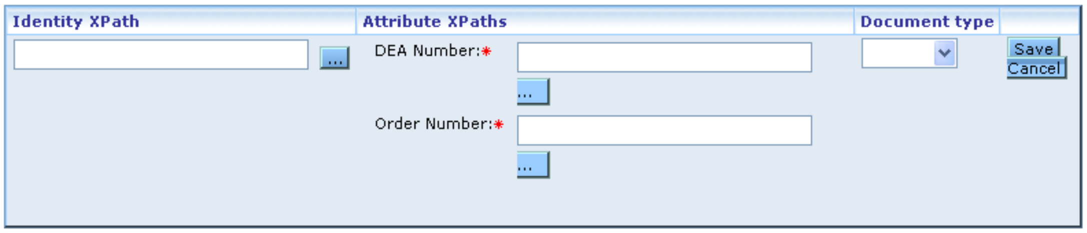


Figure 166. XPath fields for identifying CSOS document

2 Alternately, specify that all 850s are CSOS documents. This means all 850s picked up from the source named on the order sources tab are presumed to be CSOS documents. EDI documents other than 850s, however, are not tabbed for CSOS handling and proceed through normal processing without interruption.

To use this option, click **Accept all EDI 850s as CSOS purchase orders**. This displays a section (Figure 167) similar to the previous option. But the identity XPath is “accepting all EDI 850s” rather than a blank field.

If you designate all 850s as CSOS orders, you can have only one EDI document definition.

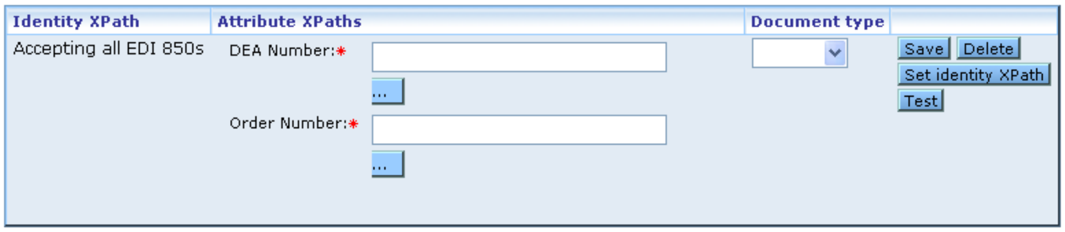


Figure 167. All 850s regarded as CSOS documents

To add XPath strings, click an ellipse (...) button to launch the XPath wizard and follow the prompts. Before you do, have a sample 850 document on your file system. The sample should be structurally identical to the actual documents to be processed. Before starting the wizard, review the sample for the data to parse. This will help you add the XPath after the wizard converts the EDI to XML.

Note: The wizard cannot verify whether a document is an 850 or another transaction type. Use only a properly formatted 850 with the wizard.

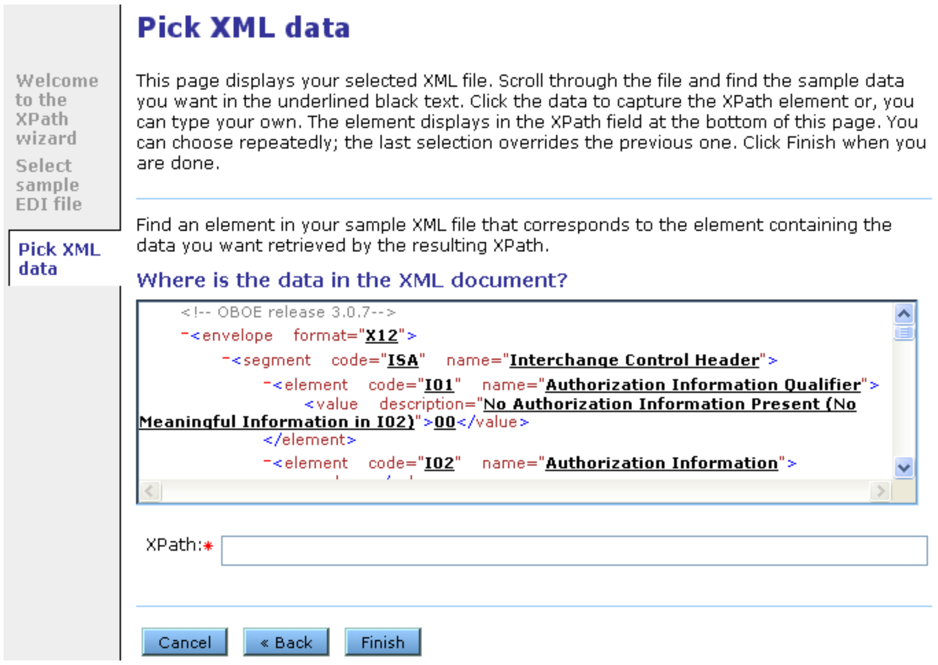
For example, in the 850 in Figure 168, we can use the value of the ST segment (850) to add the identity XPath. There are two DEA registration numbers in this document, one following the number 11 in ~N1*SU*BAXTER ACC*11*CD1234567 and the other following the number 11 in ~N1*ST*ANY DISTRIBUTOR*11*BA2893611. We want to use the DEA number belonging to the purchaser, which is the second number (the first is the supplier's DEA number). The order number is 03X123456 in the second position of the REF segment.

```
ISA*00*          *00*          *ZZ*PURCHASER      *ZZ*SUPPLIER
*030128*0647*U*00400*000002122*0*P*|~GS*PO*177667227*8006670959*20030128
*06474500*2121*X*004010~ST*850*0001~BEG*00*SA*581020016**20030128~REF*D1
*03X123456~FOB*PP~CSH*N~DTM*002*20030205~N1*SU*BAXTER
ACC*11*CD1234567~N1*ST*ANY DISTRIBUTOR*11*BA2893611~N3*1234 Any
STREET~N4*ANYTOWN
USA*WI*12345~REF*72*2,3,3N,4,5~REF*BE*F~PO1*1*28*CT***N4*10019017580~PID
*F****MORPHINE SULFATE INJ 1MG/
ML~PO4*4*25*UN~PO1*2*15*CA***N4*10019017868~PID*F****MORPHINE SULFATE
INJ 10MG/ML 1ML
AMPUL~PO4*4*25*UN~PO1*3*8*CT***N4*10019017963~PID*F****MORPHINE SULFATE
INJ 15MG/ML 20ML
MDV~PO4*25*1*UN~PO1*4*4*CA*645**N4*10019018265~PID*F****MORPHINE SULFATE
INJ 2MG/ML 1ML TUBEX
SYR~PO4*10*10*UN~PO1*5*1*FF*100**N4*10101010101~PID*F****Matches
certone.pfx~PO4*10*10*UN~CTT*5*55~SE*26*0001~GE*1*2121~IEA*1*000002122~
```

Figure 168. Sample 850 document

The XPath wizard converts the EDI into a much longer XML document. See the next topic, EDI to XML conversion example on page 522 to view the converted document.

The wizard displays the converted document on the pick XML data page (Figure 169). Using your browser's search feature, search for a value (for example, the order number 03X123456). When the value is found, click it to generate the XPath in the XPath field.



Pick XML data

This page displays your selected XML file. Scroll through the file and find the sample data you want in the underlined black text. Click the data to capture the XPath element or, you can type your own. The element displays in the XPath field at the bottom of this page. You can choose repeatedly; the last selection overrides the previous one. Click Finish when you are done.

Find an element in your sample XML file that corresponds to the element containing the data you want retrieved by the resulting XPath.

Where is the data in the XML document?

```
  <!-- OBOE release 3.0.7-->
 -<envelope  format="X12">
   -<segment  code="ISA"  name="Interchange Control Header">
     -<element  code="I01"  name="Authorization Information Qualifier">
        <value  description="No Authorization Information Present (No
Meaningful Information in I02)">00</value>
      </element>
     -<element  code="I02"  name="Authorization Information">
```

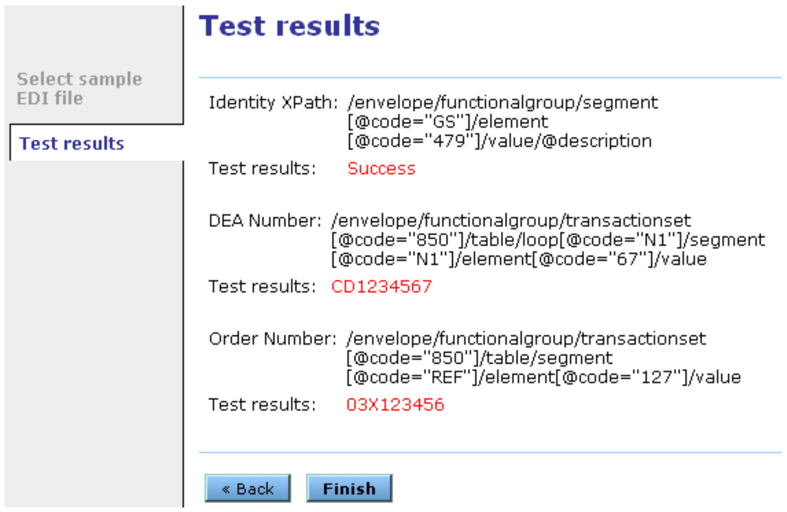
XPath:*

Figure 169. Pick XML data page in XPath wizard

Using the values desired for the sample 850 in Figure 168, we obtained the following XPaths:

Identity XPath	/envelope/functionalgroup/segment[@code="GS"]/ element[@code="479"]/value/@description
DEA number	/envelope/functionalgroup/ transactionset[@code="850"]/table/ loop[@code="N1"]/segment[@code="N1"]/ element[@code="67"]/value
Order number	/envelope/functionalgroup/ transactionset[@code="850"]/table/ segment[@code="REF"]/element[@code="127"]/ value

Click **Test** on the document types tab to check whether the XPaths are correct and yield the desired data. Figure 170 shows the test results for the preceding XPaths.



Test results

Identity XPath: /envelope/functionalgroup/segment[@code="GS"]/element[@code="479"]/value/@description

Test results: Success

DEA Number: /envelope/functionalgroup/transactionset[@code="850"]/table/loop[@code="N1"]/segment[@code="N1"]/element[@code="67"]/value

Test results: CD1234567

Order Number: /envelope/functionalgroup/transactionset[@code="850"]/table/segment[@code="REF"]/element[@code="127"]/value

Test results: 03X123456

Figure 170. XPath test results

Recall that the DEA registration number desired from the sample 850 in Figure 168 is the purchaser's number. The XPath test result, however, produced the supplier's number (CD1234567) and not the purchaser's (BA2893611). The wizard could not tell between the two N1 segments in the document and produced only the first value found. Manual editing is required to correct this. This is when knowledge of XPath is useful. If you are unfamiliar with the language, seek help from someone who knows.

The DEA number XPath added by the wizard was:

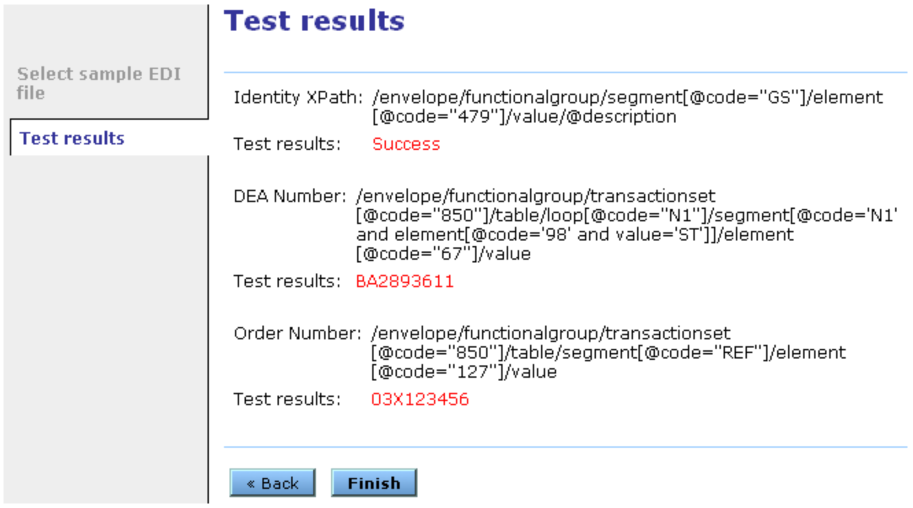
```
/envelope/functionalgroup/transactionset[@code="850"]/table/
loop[@code="N1"]/segment[@code="N1"]/element[@code="67"]/
value
```

But it must be edited as follows to produce the purchaser's DEA number:

```
/envelope/functionalgroup/transactionset[@code="850"]/table/
loop[@code="N1"]/segment[@code='N1' and element[@code='98'
and value='ST']]/element[@code="67"]/value
```

The changed part of the XPath is: **and element[@code='98' and value='ST']]**

Figure 171 shows the test results again after changing the DEA number XPath. This time the desired DEA number is produced.



Test results

Identity XPath: /envelope/functionalgroup/segment[@code="GS"]/element[@code="479"]/value/@description

Test results: Success

DEA Number: /envelope/functionalgroup/transactionset[@code="850"]/table/loop[@code="N1"]/segment[@code='N1' and element[@code='98' and value='ST']]/element[@code="67"]/value

Test results: BA2893611

Order Number: /envelope/functionalgroup/transactionset[@code="850"]/table/segment[@code="REF"]/element[@code="127"]/value

Test results: 03X123456

Figure 171. Test results after editing DEA number XPath

The last task in identifying EDI documents is to select a document type from the drop-down list. Selecting a document type associates the 850 with a stylesheet for formatting the document. On selecting to view the document in the user interface, the trading engine transforms the EDI to XML and applies the stylesheet. Before choosing a document type, however, you must add a type in Message tracker by selecting **Message tracker > Configure payload view**.

For information about adding document types with Message tracker, see Configure payload view on page 417.

Identifying XML documents

To identify XML documents for CSOS handling, click **Add an XML document type**. This displays a section (Figure 172) that has a field for an identity XPath that can be used to differentiate one type of document from another. There also are fields for specifying XPaths for the DEA registration number and order number and for selecting a document type, which is used to format a document for viewing in the user interface. You can define multiple XML document types.

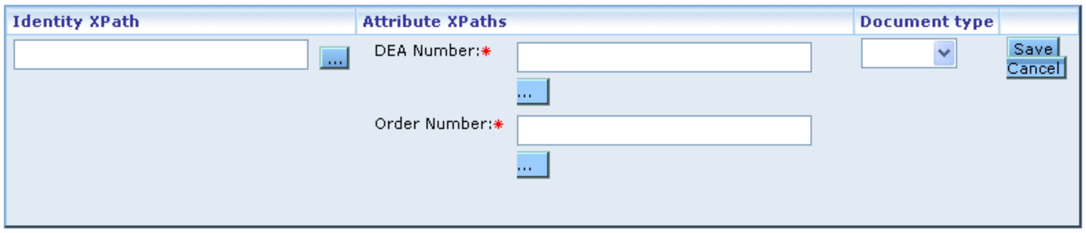


Figure 172. XPath fields for identifying CSOS document

To add XPath strings, click an ellipse (...) button to launch the XPath wizard and follow the prompts. Before you do, have a sample XML document on your file system. The sample should be structurally identical to the actual documents to be processed. Before starting the wizard, review the sample for the data to parse.

For the Identity XPath you can use any XPath for the document. This XPath serves to provide a unique identity to the document type.

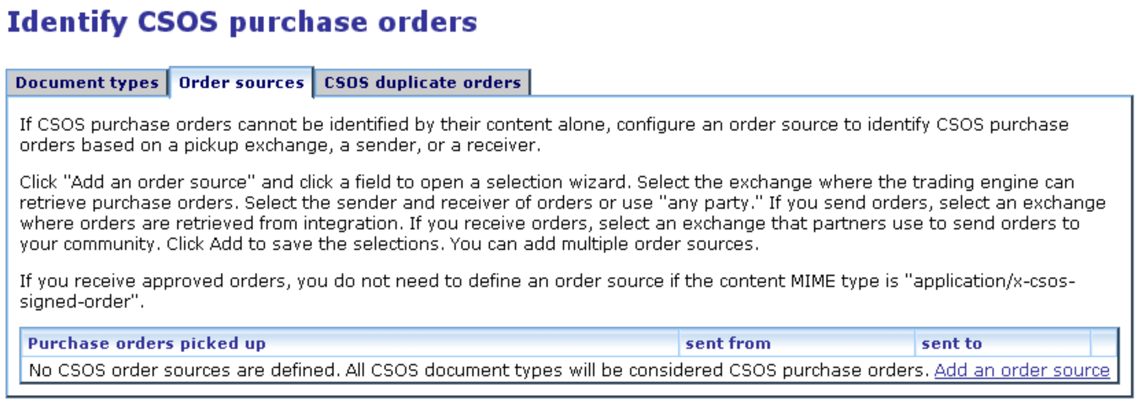
After using the wizard to add XPaths for the Identity XPath, DEA Number and Order Number fields, select a document type from the drop-down list. Selecting a document type associates the XML with a stylesheet for formatting the document. On selecting to view the document in the user interface, the trading engine applies the stylesheet. Before choosing a document type, however, you must add a type in Message tracker by selecting **Message tracker > Configure payload view**.

For information about adding document types with Message tracker, see Configure payload view on page 417.

Order sources tab

The order sources tab of the identify CSOS purchase orders page lets you set filtering conditions in addition to those on the document types tab.

Whether you should use the order sources tab depends on your situation. For instance, if you receive CSOS orders via certain message protocols — AS1, AS2, Secure file, Secure e-mail, ebXML — a content MIME type of application/x-csos-signed-order is included in inbound message headers. This triggers the trading engine to validate and unpack the inbound messages as CSOS documents, and using the order sources tab is unnecessary. On the other hand, if you send CSOS orders, you can use the order sources tab to specify a particular integration exchange for picking up documents identified on the document types tab. For instance, you may want to specify one integration pickup for CSOS EDI 850 documents if your company also sends non-CSOS EDI 850 documents.



Identify CSOS purchase orders

Document types | Order sources | CSOS duplicate orders

If CSOS purchase orders cannot be identified by their content alone, configure an order source to identify CSOS purchase orders based on a pickup exchange, a sender, or a receiver.

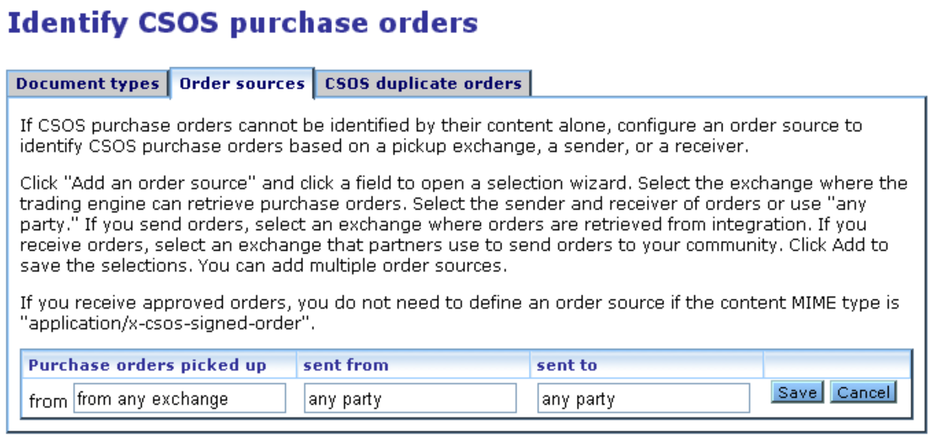
Click "Add an order source" and click a field to open a selection wizard. Select the exchange where the trading engine can retrieve purchase orders. Select the sender and receiver of orders or use "any party." If you send orders, select an exchange where orders are retrieved from integration. If you receive orders, select an exchange that partners use to send orders to your community. Click Add to save the selections. You can add multiple order sources.

If you receive approved orders, you do not need to define an order source if the content MIME type is "application/x-csos-signed-order".

Purchase orders picked up	sent from	sent to	
No CSOS order sources are defined. All CSOS document types will be considered CSOS purchase orders. Add an order source			

Figure 173. Order sources tab of Identify CSOS purchase orders page

Click **Add an order source** at the bottom of the page. The page displays an available default source of **any exchange** from **any party** to **any party**. This is selected only if you click the **Save** button.



Identify CSOS purchase orders

Document types | Order sources | CSOS duplicate orders

If CSOS purchase orders cannot be identified by their content alone, configure an order source to identify CSOS purchase orders based on a pickup exchange, a sender, or a receiver.

Click "Add an order source" and click a field to open a selection wizard. Select the exchange where the trading engine can retrieve purchase orders. Select the sender and receiver of orders or use "any party." If you send orders, select an exchange where orders are retrieved from integration. If you receive orders, select an exchange that partners use to send orders to your community. Click Add to save the selections. You can add multiple order sources.

If you receive approved orders, you do not need to define an order source if the content MIME type is "application/x-csos-signed-order".

Purchase orders picked up	sent from	sent to	
from from any exchange	any party	any party	Save Cancel

Figure 174. Default order sources settings

This default choice means:

If sending orders

All purchase order documents defined on the document types tab are picked up from any integration exchange and queued for signing.

If receiving orders

All purchase orders defined on the document types tab and received from any partner are unpacked, validated against the root certificate for the user’s signature and DEA registration number, and routed to an integration delivery exchange.

The default setting may be too broad for many users. You may want to use more specific conditions.

To make a selection, click on the **Purchase order picked up**, **sent from** or **sent to** field on the CSOS orders sources page. This opens a wizard that lets you choose delivery exchanges (inbound or outbound transports) and parties (communities or partners). The following table shows example configurations.

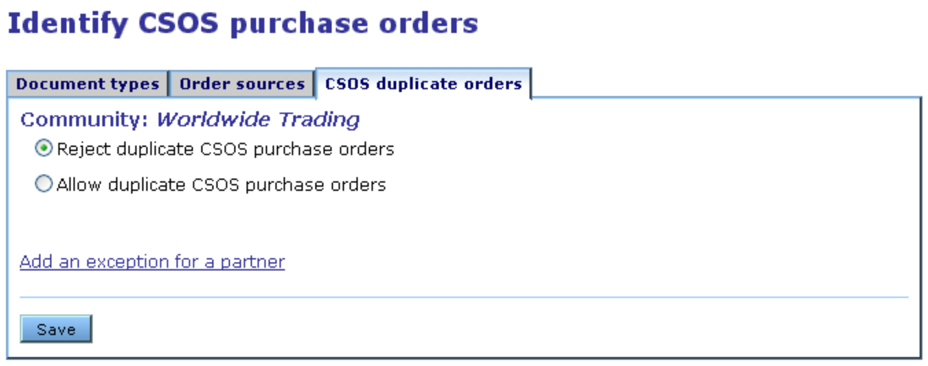
Purchase orders picked up from	sent from	sent to
file system integration	Your community	Partner A
file system integration	Your community	Any partner
Trading transport for receiving messages from partners	Partner A	Your community
Trading transport for receiving messages from partners	Any partner	Your community

Once you have made your selections, click **Save**. You can specify multiple sources, but take care not to set up overlapping conditions. Once added, you can change or delete a source by using the appropriate buttons.

When sending orders, purchase orders are plucked from the normal processing routine of the trading engine and queued for signing. Once signed, the messages are placed back in the trading engine flow. For example, if you choose file system integration pickup as the message source and any party as the sender and receiver, all properly formatted purchase orders defined on the document types tab are queued for signing. All other EDI documents and document types (XML and binary) are ignored. Once signed, the orders are placed back in the trading engine flow for packaging and sending to partners.

CSOS duplicate orders tab

The CSOS duplicate orders tab of the identify CSOS purchase orders page (Figure 175) lets you specify whether to accept or reject duplicate CSOS orders.


Identify CSOS purchase orders

Community: *Worldwide Trading*

[✓ Reject duplicate CSOS purchase orders]

Allow duplicate CSOS purchase orders

Figure 175. CSOS duplicate orders tab

CSOS duplicate checking is done for CSOS purchase orders your community receives from partners. If the community is a WebTrader sponsor, documents received from its WebTrader partners also are checked. Duplicate checking is not done for orders picked up from integration, unless the community that picks up the order is the named receiver (a rare case).

If your community only sends signed orders but does not receive any, duplicate checking is not applicable and you can ignore this tab.

The target documents are the EDI or XML documents defined on the document types tab of the identify CSOS purchase orders page.

By default, the radio button for rejecting duplicate purchase orders is selected on the CSOS duplicate orders tab. When selected, the trading engine checks whether an order duplicates a previously received order in the following ways:

- Duplicate order number
- Duplicate DEA registration number
- Duplicate sender name
- Duplicate receiver name

If all of these values are the same as those in a previously received order, the document is given a failed status. You can search for failed documents in Message tracker.

If you also have configured the trading engine to check for duplicate EDI documents, checking for duplicate CSOS orders is performed in addition to the duplicate EDI checking.

Aside from choosing whether a community can allow or reject duplicates, you can set up exceptions to the selected behavior on a per-partner basis.

Whether you choose to reject or allow duplicate orders, the choice applies to all orders received for the community, unless you define partner-specific exceptions.

To add a partner-specific exception to the selected behavior, click **Add an exception for a partner**. A wizard displays to help you locate and select the partner or partners you wish to add. You can add multiple partners to the exception list at one time using the wizard.

The trading engine applies the opposite of the selected behavior to any partners that display in the exception list.

Note that the CSOS duplicate order tabs can be found on two pages in the user interface. One location is the identify CSOS purchase orders page at **CSOS** > **Configure CSOS**. The other is the message validation rules page, which is opened by clicking **Message validation** on the navigation graphic at the top of a community summary page. To configure CSOS duplicate checking, you only need to use one of these pages.

Sign pending orders

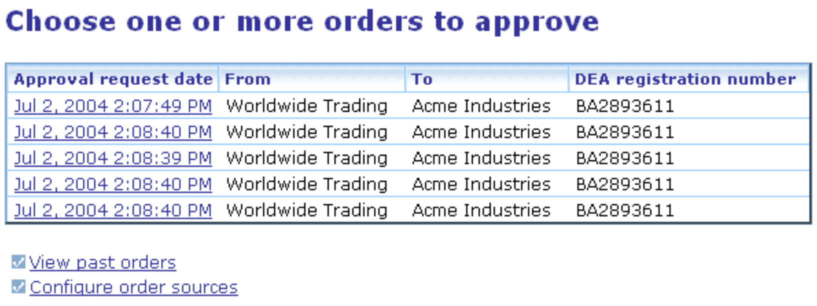
Use this procedure to digitally sign CSOS orders and release them so the trading engine can package and send them to a partner. This procedure presumes all CSOS configuration steps have been completed correctly.

Steps

1 Log on to the user interface and check whether any CSOS orders are awaiting signature.

The system generates an alert when CSOS orders are awaiting signature. You also can check for pending orders by selecting **CSOS > View pending orders** on the toolbar.

Any orders in the signing queue are listed on the orders to approve page (Figure 176).

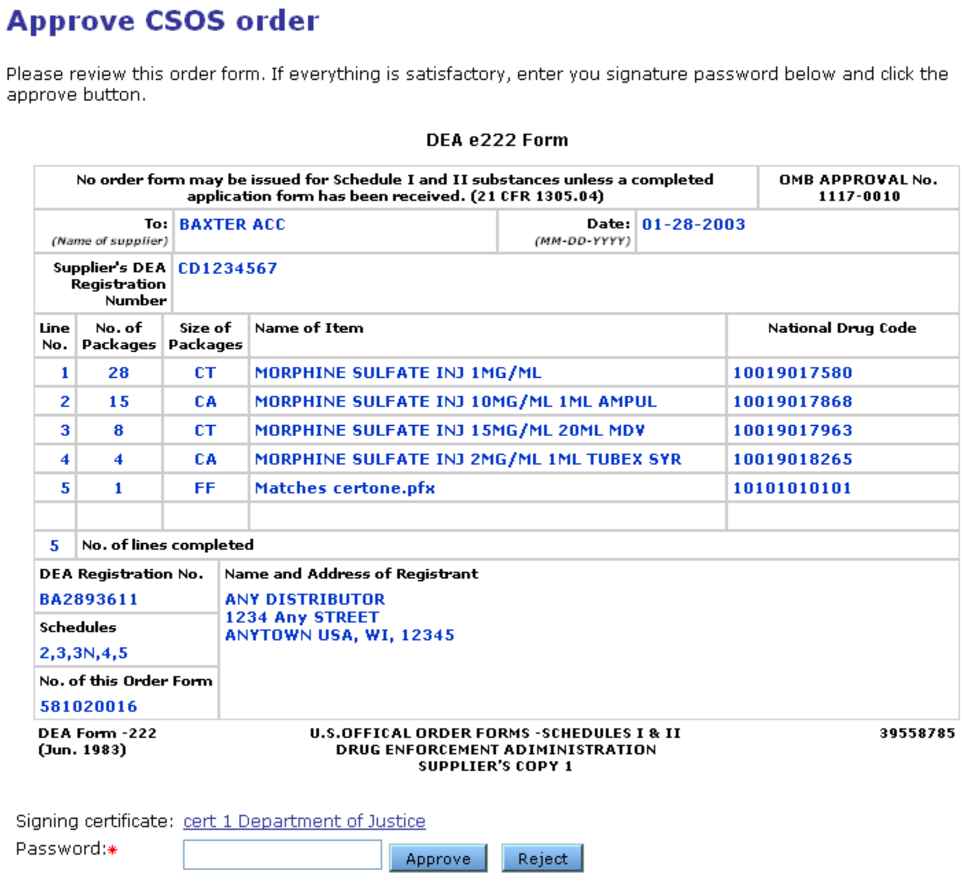


Choose one or more orders to approve

Approval request date	From	To	DEA registration number
Jul 2, 2004 2:07:49 PM	Worldwide Trading	Acme Industries	BA2893611
Jul 2, 2004 2:08:40 PM	Worldwide Trading	Acme Industries	BA2893611
Jul 2, 2004 2:08:39 PM	Worldwide Trading	Acme Industries	BA2893611
Jul 2, 2004 2:08:40 PM	Worldwide Trading	Acme Industries	BA2893611
Jul 2, 2004 2:08:40 PM	Worldwide Trading	Acme Industries	BA2893611

Figure 176. CSOS orders to approve page

2 Click the date of a pending order to display it on the approve CSOS order page (Figure 177).



Approve CSOS order

Please review this order form. If everything is satisfactory, enter you signature password below and click the approve button.

DEA e222 Form

No order form may be issued for Schedule I and II substances unless a completed application form has been received. (21 CFR 1305.04)				OMB APPROVAL No. 1117-0010
To: (Name of supplier)	BAXTER ACC		Date: (MM-DD-YYYY)	01-28-2003
Supplier's DEA Registration Number	CD1234567			

Line No.	No. of Packages	Size of Packages	Name of Item	National Drug Code
1	28	CT	MORPHINE SULFATE INJ 1MG/ML	10019017580
2	15	CA	MORPHINE SULFATE INJ 10MG/ML 1ML AMPUL	10019017868
3	8	CT	MORPHINE SULFATE INJ 15MG/ML 20ML MDV	10019017963
4	4	CA	MORPHINE SULFATE INJ 2MG/ML 1ML TUBEX SYR	10019018265
5	1	FF	Matches certone.pfx	10101010101
5	No. of lines completed			

DEA Registration No. BA2893611	Name and Address of Registrant ANY DISTRIBUTOR 1234 Any STREET ANYTOWN USA, WI, 12345
Schedules 2,3,3N,4,5	
No. of this Order Form 581020016	

DEA Form -222 (Jun. 1983) U.S.OFFICAL ORDER FORMS -SCHEDULES I & II DRUG ENFORCEMENT ADIMINISTRATION SUPPLIER'S COPY 1 39558785

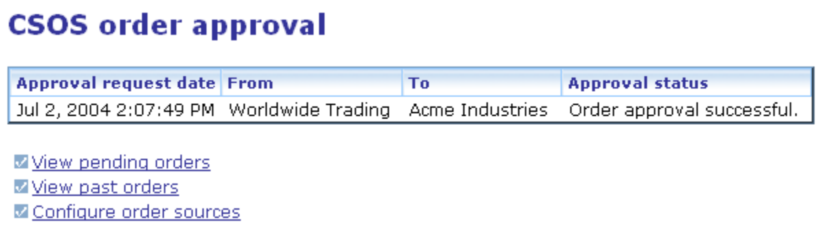
Signing certificate: cert 1 Department of Justice
Password:*

Figure 177. Approve CSOS order page

3 Review the order. If the document is satisfactory, type your password in the field at the bottom of the page and click **Approve**. The password is the same one you use to log on to the user interface. If you enter an incorrect password, the system prompts you to try again.

Once you click **Approve**, the CSOS order approval page displays. The approval status column shows the order approval was successful (Figure 178). The trading engine resumes normal outbound processing of the document.

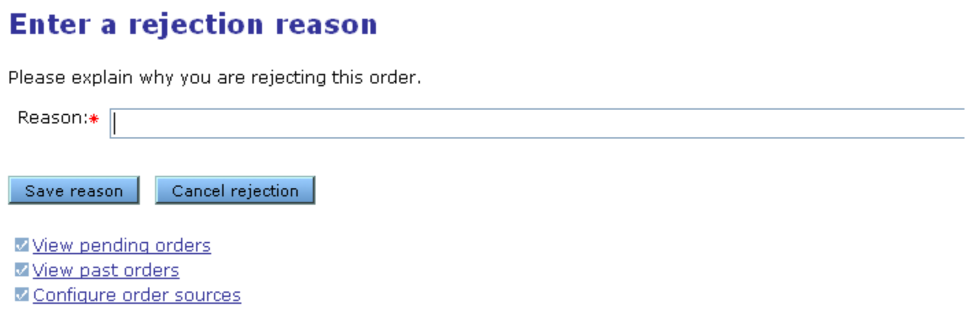
If you have not already added a user signing certificate, a message prompting you to do so is displayed in place of the password field.


CSOS order approval

Approval request date	From	To	Approval status
Jul 2, 2004 2:07:49 PM	Worldwide Trading	Acme Industries	Order approval successful.

Figure 178. CSOS order approval page

If the document is unsatisfactory, click **Reject**. This action opens a page where you can type a reason for rejecting the document (Figure 179). Click **Save reason** when you are done. A rejected CSOS order is reported in Message tracker as a failed message. The failure reason you type is displayed in the message details for the rejected order.


Enter a rejection reason

Please explain why you are rejecting this order.

Reason:*

Figure 179. Enter a reject reason page

4 You have several options for continuing:

Click **View pending orders** to approve another order in the queue.

or

Click **View past orders** to go to Message tracker and follow the document's trading status.

the next topic, EDI to XML conversion example

The user interface's XPath wizard dynamically converts EDI documents to XML for the purpose of letting users select XPath strings for identifying values for parsing. For an explanation of why this occurs, see Identifying EDI documents on page 509.

Figure 180 is an example of an 850 document in pre-converted EDI format.

```
ISA*00*          *00*          *ZZ*PURCHASER      *ZZ*SUPPLIER
*030128*0647*U*00400*000002122*0*P*|~GS*PO*177667227*8006670959*20030128
*06474500*2121*X*004010~ST*850*0001~BEG*00*SA*581020016**20030128~REF*D1
*03X123456~FOB*PP~CSH*N~DTM*002*20030205~N1*SU*BAXTER
ACC*11*CD1234567~N1*ST*ANY DISTRIBUTOR*11*BA2893611~N3*1234 Any
STREET~N4*ANYTOWN
USA*WI*12345~REF*72*2,3,3N,4,5~REF*BE*F~PO1*1*28*CT***N4*10019017580~PID
*F****MORPHINE SULFATE INJ 1MG/
ML~PO4*4*25*UN~PO1*2*15*CA***N4*10019017868~PID*F****MORPHINE SULFATE
INJ 10MG/ML 1ML
AMPUL~PO4*4*25*UN~PO1*3*8*CT***N4*10019017963~PID*F****MORPHINE SULFATE
INJ 15MG/ML 20ML
MDV~PO4*25*1*UN~PO1*4*4*CA*645**N4*10019018265~PID*F****MORPHINE SULFATE
INJ 2MG/ML 1ML TUBEX
SYR~PO4*10*10*UN~PO1*5*1*FF*100**N4*10101010101~PID*F****Matches
certone.pfx~PO4*10*10*UN~CTT*5*55~SE*26*0001~GE*1*2121~IEA*1*000002122~
```

Figure 180. Sample 850 document

The following is the same document after the XPath wizard converted it to XML:

```
<!-- OBOE release 3.0.7-->
-<envelopeformat="X12">
-<segmentcode="ISA"name="Interchange Control Header">
-<elementcode="I01"name="Authorization Information Qualifier">
<valuedescription="No Authorization Information Present (No Meaningful
Information in


I02)">00</value>
</element>
-<elementcode="I02"name="Authorization Information">
<value></value>
</element>
-<elementcode="I03"name="Security Information Qualifier">
<valuedescription="No Security Information Present (No Meaningful
Information in


I04)">00</value>
</element>
-<elementcode="I04"name="Security Information">
<value></value>
</element>
```

```
-<elementcode="I05"name="Interchange ID Qualifier">
<valuedescription="Mutually Defined">ZZ</value>
</element>
-<elementcode="I06"name="Interchange Sender ID">
<value>PURCHASER </value>
</element>
-<elementcode="I05"name="Interchange ID Qualifier">
<valuedescription="Mutually Defined">ZZ</value>
</element>
-<elementcode="I07"name="Interchange Receiver ID">
<value>SUPPLIER </value>
</element>
-<elementcode="I08"name="Interchange Date">
<value>030128</value>
</element>
-<elementcode="I09"name="Interchange Time">
<value>0647</value>
</element>
-<elementcode="I10"name="Interchange Control Standards Identifier">
<valuedescription="U.S. EDI Community of ASC X12, TDCC, and UCS">U</value>
</element>
-<elementcode="I11"name="Interchange Control Version Number">
<valuedescription="Standard Issued as ANSI X12.5-1997">00400</value>
</element>
-<elementcode="I12"name="Interchange Control Number">
<value>000,002,122</value>
</element>
-<elementcode="I13"name="Acknowledgment Requested">
<valuedescription="No Acknowledgment Requested">0</value>
</element>
-<elementcode="I14"name="Usage Indicator">
<valuedescription="Production Data">P</value>
</element>
-<elementcode="I15"name="Component Element Separator">
<value>|</value>
</element>

-<functionalgroup>
-<segmentcode="GS"name="Functional Group Header">
-<elementcode="479"name="Functional Identifier Code">
<valuedescription="Purchase Order (850)">PO</value>
</element>
-<elementcode="142"name="Application Sender's Code">
<value>177667227</value>
</element>
-<elementcode="124"name="Application Receiver's Code">
<value>8006670959</value>
</element>
-<elementcode="373"name="Date">
<value>20030128</value>
</element>
-<elementcode="337"name="Time">
<value>064745</value>
</element>
-<elementcode="28"name="Group Control Number">
<value>2,121</value>
</element>
-<elementcode="455"name="Responsible Agency Code">
<valuedescription="Accredited Standards Committee X12">X</value>
</element>
-<elementcode="480"name="Version / Release / Industry Identifier Code">
<valuedescription="Draft Standards Approved for Publication by ASC X12
Procedures Review Board
```

```
through October 1997">004010</value>
</element>

-<transactionsetcode="850"name="Purchase Order">
-<tablesection="header">
-<segmentcode="ST"name="Transaction Set Header">
-<elementcode="143"name="Transaction Set Identifier Code">
<valuedescription="Purchase Order">850</value>
</element>
-<elementcode="329"name="Transaction Set Control Number">
<value>0001</value>
</element>
</segment>
-<segmentcode="BEG"name="Beginning Segment for Purchase Order">
-<elementcode="353"name="Transaction Set Purpose Code">
<valuedescription="Original">00</value>
</element>
-<elementcode="92"name="Purchase Order Type Code">
<valuedescription="Stand-alone Order">SA</value>
</element>
-<elementcode="324"name="Purchase Order Number">
<value>581020016</value>
</element>
-<elementcode="328"name="Release Number">
<value/>
</element>
-<elementcode="373"name="Date">
<value>20030128</value>
</element>
</segment>
-<segmentcode="REF"name="Reference Identification">
-<elementcode="128"name="Reference Identification Qualifier">
<valuedescription="Drug Enforcement Administration Order Blank Number">D1</
value>
</element>
-<elementcode="127"name="Reference Identification">
<value>03X123456</value>
</element>
</segment>
-<segmentcode="FOB"name="F.O.B. Related Instructions">
-<elementcode="146"name="Shipment Method of Payment">
<valuedescription="Prepaid (by Seller)">PP</value>
</element>
</segment>
-<segmentcode="CSH"name="Sales Requirements">
-<elementcode="563"name="Sales Requirement Code">
<valuedescription="No Back Order">N</value>
</element>
</segment>
-<segmentcode="DTM"name="Date/Time Reference">
-<elementcode="374"name="Date/Time Qualifier">
<valuedescription="Delivery Requested">002</value>
</element>
-<elementcode="373"name="Date">
<value>20030205</value>
</element>
</segment>
-<loopcode="N1"name="Name">
-<segmentcode="N1"name="Name">
-<elementcode="98"name="Entity Identifier Code">
<valuedescription="Supplier/Manufacturer">SU</value>
</element>
-<elementcode="93"name="Name">
```

```
<value>BAXTER ACC</value>
</element>
-<elementcode="66"name="Identification Code Qualifier">
<valuedescription="Drug Enforcement Administration (DEA)">11</value>
</element>
-<elementcode="67"name="Identification Code">
<value>CD1234567</value>
</element>

</loop>
-<loopcode="N1"name="Name">
-<segmentcode="N1"name="Name">
-<elementcode="98"name="Entity Identifier Code">
<valuedescription="Ship To">ST</value>
</element>
-<elementcode="93"name="Name">
<value>ANY DISTRIBUTOR</value>
</element>
-<elementcode="66"name="Identification Code Qualifier">
<valuedescription="Drug Enforcement Administration (DEA)">11</value>
</element>
-<elementcode="67"name="Identification Code">
<value>BA2893611</value>
</element>
</segment>
-<segmentcode="N3"name="Address Information">
-<elementcode="166"name="Address Information">
<value>1234 Any STREET</value>
</element>
</segment>
-<segmentcode="N4"name="Geographic Location">
-<elementcode="19"name="City Name">
<value>ANYTOWN USA</value>
</element>
-<elementcode="156"name="State or Province Code">
<valuedescription="WI">WI</value>
</element>
-<elementcode="116"name="Postal Code">
<valuedescription="12345">12345</value>
</element>
</segment>
-<segmentcode="REF"name="Reference Identification">
-<elementcode="128"name="Reference Identification Qualifier">
<valuedescription="Schedule Reference Number">72</value>
</element>
-<elementcode="127"name="Reference Identification">
<value>2,3,3N,4,5</value>
</element>
</segment>
-<segmentcode="REF"name="Reference Identification">
-<elementcode="128"name="Reference Identification Qualifier">
<valuedescription="Business Activity">BE</value>
</element>
-<elementcode="127"name="Reference Identification">
<value>F</value>
</element>
</segment>
</loop>
</table>
-<tablesection="detail">
-<loopcode="PO1"name="Baseline Item Data">
-<segmentcode="PO1"name="Baseline Item Data">
-<elementcode="350"name="Assigned Identification">
<value>1</value>
```

```
</element>
-<elementcode="330"name="Quantity Ordered">
<value>28</value>
</element>
-<elementcode="355"name="Unit or Basis for Measurement Code">
<valuedescription="Carton">CT</value>
</element>
-<elementcode="212"name="Unit Price">
<value/>
</element>
-<elementcode="639"name="Basis of Unit Price Code">
<valuedescription=""/>
</element>
-<elementcode="235"name="Product/Service ID Qualifier">
<valuedescription="National Drug Code in 5-4-2 Format">N4</value>
</element>
-<elementcode="234"name="Product/Service ID">
<value>10019017580</value>
</element>

-<loopcode="PID"name="Product/Item Description">
-<segmentcode="PID"name="Product/Item Description">
-<elementcode="349"name="Item Description Type">
<valuedescription="Free-form">F</value>
</element>
-<elementcode="750"name="Product/Process Characteristic Code">
<valuedescription=""/>
</element>
-<elementcode="559"name="Agency Qualifier Code">
<valuedescription=""/>
</element>
-<elementcode="751"name="Product Description Code">
<value/>
</element>
-<elementcode="352"name="Description">
<value>MORPHINE SULFATE INJ 1MG/ML</value>
</element>
</segment>
</loop>
-<segmentcode="PO4"name="Item Physical Details">
-<elementcode="356"name="Pack">
<value>4</value>
</element>
-<elementcode="357"name="Size">
<value>25</value>
</element>
-<elementcode="355"name="Unit or Basis for Measurement Code">
<valuedescription="Unit">UN</value>
</element>
</segment>
</loop>
-<loopcode="PO1"name="Baseline Item Data">
-<segmentcode="PO1"name="Baseline Item Data">
-<elementcode="350"name="Assigned Identification">
<value>2</value>
</element>
-<elementcode="330"name="Quantity Ordered">
<value>15</value>
</element>
-<elementcode="355"name="Unit or Basis for Measurement Code">
<valuedescription="Case">CA</value>
</element>
-<elementcode="212"name="Unit Price">
<value/>
```

```
</element>
-<elementcode="639"name="Basis of Unit Price Code">
<valuedescription=""/>
</element>
-<elementcode="235"name="Product/Service ID Qualifier">
<valuedescription="National Drug Code in 5-4-2 Format">N4</value>
</element>
-<elementcode="234"name="Product/Service ID">
<value>10019017868</value>
</element>

-<loopcode="PID"name="Product/Item Description">
-<segmentcode="PID"name="Product/Item Description">
-<elementcode="349"name="Item Description Type">
<valuedescription="Free-form">F</value>
</element>
-<elementcode="750"name="Product/Process Characteristic Code">
<valuedescription=""/>
</element>
-<elementcode="559"name="Agency Qualifier Code">
<valuedescription=""/>
</element>
-<elementcode="751"name="Product Description Code">
<value/>
</element>
-<elementcode="352"name="Description">
<value>MORPHINE SULFATE INJ 10MG/ML 1ML AMPUL</value>
</element>
</segment>
</loop>
-<segmentcode="PO4"name="Item Physical Details">
-<elementcode="356"name="Pack">
<value>4</value>
</element>
-<elementcode="357"name="Size">
<value>25</value>
</element>
-<elementcode="355"name="Unit or Basis for Measurement Code">
<valuedescription="Unit">UN</value>
</element>
</segment>
</loop>
-<loopcode="PO1"name="Baseline Item Data">
-<segmentcode="PO1"name="Baseline Item Data">
-<elementcode="350"name="Assigned Identification">
<value>3</value>
</element>
-<elementcode="330"name="Quantity Ordered">
<value>8</value>
</element>
-<elementcode="355"name="Unit or Basis for Measurement Code">
<valuedescription="Carton">CT</value>
</element>
-<elementcode="212"name="Unit Price">
<value/>
</element>
-<elementcode="639"name="Basis of Unit Price Code">
<valuedescription=""/>
</element>
-<elementcode="235"name="Product/Service ID Qualifier">
<valuedescription="National Drug Code in 5-4-2 Format">N4</value>
</element>
-<elementcode="234"name="Product/Service ID">
<value>10019017963</value>
```

```
</element>

-<loopcode="PID"name="Product/Item Description">
-<segmentcode="PID"name="Product/Item Description">
-<elementcode="349"name="Item Description Type">
<valuedescription="Free-form">F</value>
</element>
-<elementcode="750"name="Product/Process Characteristic Code">
<valuedescription=""/>
</element>
-<elementcode="559"name="Agency Qualifier Code">
<valuedescription=""/>
</element>
-<elementcode="751"name="Product Description Code">
<value/>
</element>
-<elementcode="352"name="Description">
<value>MORPHINE SULFATE INJ 15MG/ML 20ML MDV</value>
</element>
</segment>
</loop>
-<segmentcode="PO4"name="Item Physical Details">
-<elementcode="356"name="Pack">
<value>25</value>
</element>
-<elementcode="357"name="Size">
<value>1</value>
</element>
-<elementcode="355"name="Unit or Basis for Measurement Code">
<valuedescription="Unit">UN</value>
</element>
</segment>
</loop>
-<loopcode="PO1"name="Baseline Item Data">
-<segmentcode="PO1"name="Baseline Item Data">
-<elementcode="350"name="Assigned Identification">
<value>4</value>
</element>
-<elementcode="330"name="Quantity Ordered">
<value>4</value>
</element>
-<elementcode="355"name="Unit or Basis for Measurement Code">
<valuedescription="Case">CA</value>
</element>
-<elementcode="212"name="Unit Price">
<value>645</value>
</element>
-<elementcode="639"name="Basis of Unit Price Code">
<valuedescription=""/>
</element>
-<elementcode="235"name="Product/Service ID Qualifier">
<valuedescription="National Drug Code in 5-4-2 Format">N4</value>
</element>
-<elementcode="234"name="Product/Service ID">
<value>10019018265</value>
</element>
</segment>
-<loopcode="PID"name="Product/Item Description">
-<segmentcode="PID"name="Product/Item Description">
-<elementcode="349"name="Item Description Type">
<valuedescription="Free-form">F</value>
</element>
-<elementcode="750"name="Product/Process Characteristic Code">
<valuedescription=""/>
```

```
</element>
-<elementcode="559"name="Agency Qualifier Code">
<valuedescription=""/>
</element>
-<elementcode="751"name="Product Description Code">
<value/>
</element>
-<elementcode="352"name="Description">
<value>MORPHINE SULFATE INJ 2MG/ML 1ML TUBEX SYR</value>
</element>

</loop>
-<segmentcode="PO4"name="Item Physical Details">
-<elementcode="356"name="Pack">
<value>10</value>
</element>
-<elementcode="357"name="Size">
<value>10</value>
</element>
-<elementcode="355"name="Unit or Basis for Measurement Code">
<valuedescription="Unit">UN</value>
</element>
</segment>
</loop>
-<loopcode="PO1"name="Baseline Item Data">
-<segmentcode="PO1"name="Baseline Item Data">
-<elementcode="350"name="Assigned Identification">
<value>5</value>
</element>
-<elementcode="330"name="Quantity Ordered">
<value>1</value>
</element>
-<elementcode="355"name="Unit or Basis for Measurement Code">
<valuedescription="Hundred Cubic Meters">FF</value>
</element>
-<elementcode="212"name="Unit Price">
<value>100</value>
</element>
-<elementcode="639"name="Basis of Unit Price Code">
<valuedescription=""/>
</element>
-<elementcode="235"name="Product/Service ID Qualifier">
<valuedescription="National Drug Code in 5-4-2 Format">N4</value>
</element>
-<elementcode="234"name="Product/Service ID">
<value>10101010101</value>
</element>
</segment>
-<loopcode="PID"name="Product/Item Description">
-<segmentcode="PID"name="Product/Item Description">
-<elementcode="349"name="Item Description Type">
<valuedescription="Free-form">F</value>
</element>
-<elementcode="750"name="Product/Process Characteristic Code">
<valuedescription=""/>
</element>
-<elementcode="559"name="Agency Qualifier Code">
<valuedescription=""/>
</element>
-<elementcode="751"name="Product Description Code">
<value/>
</element>
-<elementcode="352"name="Description">
<value>Matches certone.pfx</value>
```

```
</element>
</segment>
</loop>
-<segmentcode="PO4"name="Item Physical Details">
-<elementcode="356"name="Pack">
<value>10</value>
</element>
-<elementcode="357"name="Size">
<value>10</value>
</element>
-<elementcode="355"name="Unit or Basis for Measurement Code">
<valuedescription="Unit">UN</value>
</element>
</segment>
</loop>
</table>
-<tablesection="summary">
-<loopcode="CTT"name="Transaction Totals">
-<segmentcode="CTT"name="Transaction Totals">
-<elementcode="354"name="Number of Line Items">
<value>5</value>
</element>
-<elementcode="347"name="Hash Total">
<value>55</value>
</element>
</segment>
</loop>
-<segmentcode="SE"name="Transaction Set Trailer">
-<elementcode="96"name="Number of Included Segments">
<value>26</value>
</element>
-<elementcode="329"name="Transaction Set Control Number">
<value>0001</value>
</element>
</segment>
</table>
</transactionset>
-<segmentcode="GE"name="Functional Group Trailer">
-<elementcode="97"name="Number of Transaction Sets Included">
<value>1</value>
</element>
-<elementcode="28"name="Group Control Number">
<value>2,121</value>
</element>
</segment>
</functionalgroup>
-<segmentcode="IEA"name="Interchange Control Trailer">
-<elementcode="I16"name="Number of Included Functional Groups">
<value>1</value>
</element>
-<elementcode="I12"name="Interchange Control Number">
<value>000,002,122</value>
</element>
</segment>
</envelope>
```

❖ ❖ ❖

Index