



# **Administrators' Guide**

Synchrony EndPoint Activator

Version 5.5.1

August 24, 2007



Copyright © Axway Software, 2007  
All rights reserved.

This documentation describes the following Axway software:  
Synchrony EndPoint Activator 5.5.1

No part of this publication may be reproduced, transmitted, stored in a retrieval system, or translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of the copyright owner, Axway Software.

This document, provided for informational purposes only, may be subject to significant modification. The descriptions and information in this document may not necessarily accurately represent or reflect the current or planned functionalities of this product. Axway Software may change this publication, the product described herein, or both. These changes will be incorporated in new versions of this document. Axway Software does not warrant that this document is error free.

Axway Software recognizes the rights of the holders of all trademarks used in its publications.

For a list of third-party files included in this product, see **ACKNOWLEDGMENTS.pdf** in **[install directory]\[build number]\Licenses**.

The documentation may provide hyperlinks to third-party web sites or access to third-party content. Links and access to these sites are provided for your convenience only. Axway does not control, endorse or guarantee content found in such sites. Axway is not responsible for any content, associated links, resources or services associated with a third-party site. Axway shall not be liable for any loss or damage of any sort associated with your use of third-party content.

---

# Contents

User comments .....	xiii
<b>Chapter 1. System requirements</b>	<b>1</b>
Hardware .....	1
Windows and UNIX computers .....	1
E-mail server .....	2
Software .....	3
Microsoft Windows.....	3
UNIX .....	4
OS patches and JRE .....	5
Database.....	6
Internet connection .....	6
Internet browser .....	7
E-mail clients.....	8
E-mail service provider .....	8
Remote access .....	8
Adobe Acrobat.....	9
Port assignments and possible conflicts .....	9
Security considerations .....	11
About the user license file .....	12
Technical support .....	12
<b>Chapter 2. Installing and starting the server</b>	<b>15</b>
Installation outline .....	15
Before installing .....	17
Guidelines for installing on UNIX .....	17
Start the server on Windows .....	18
Start the server on UNIX .....	19
Run as a Windows service .....	19
Install service.....	20
Uninstall service .....	21
Use postInstall to add Windows service.....	21
Open the user interface .....	22
Before logging on the first time.....	22
Log on procedure .....	22
Configuring external SMTP server .....	23
Restarting for server performance .....	23
Stop the server .....	24
Uninstalling .....	24
<b>Chapter 3. Upgrading</b>	<b>25</b>
Directory tree's role in upgrading .....	25

---

Upgrading from version 5.1 or later .....	26
Back up database .....	26
Back up common directory .....	27
Carry functionality forward .....	27
Note for ebXML users .....	28
Upgrade steps .....	28
Change database from Sybase to Derby .....	32
Configure properties file .....	32
Running Data Mover .....	34
File comparison tools for upgrades .....	36
Migrating version 4.2 trading profiles .....	39
Export all profiles .....	39
Import all profiles .....	41
Post-import tasks .....	41
<b>Chapter 4. Tools and options</b>	<b>43</b>
Tools in bin directory .....	43
Tools in tools directory .....	45
Installation directory tree .....	49
[install directory]\[build number] .....	50
[install directory]\common .....	51
Database configuration tool .....	52
Binding network interfaces .....	53
Single sign-on interface .....	54
Planning for upgrades and disaster recovery .....	56
Running the UI over HTTPS .....	58
Configure HTTPS .....	58
Switch between HTTP and HTTPS .....	60
<b>Chapter 5. Navigating the user interface</b>	<b>63</b>
The toolbar .....	63
Navigation aids for the trading engine .....	65
Help for users of earlier versions .....	65
Relating 4.2.x company profile to new system .....	66
Relating 4.2.x partner profile to new system .....	69
<b>Chapter 6. UI usage with proxy servers</b>	<b>73</b>
Deployment in proxy environment .....	73
Forward proxy .....	73
Reverse proxy .....	74
<b>Chapter 7. User administration</b>	<b>77</b>
The admin user .....	78
Add, change, delete users .....	78
Change password .....	79
Add, change, delete roles .....	79
Role permissions .....	80

---

Partner restrictions for roles .....	83
Managing multiple partner-restricted roles .....	84
Global user settings .....	85
Session management tab .....	85
User security tab .....	86
Unlocking a blocked user .....	88
<b>Chapter 8. Activity tracking and logs</b>	<b>89</b>
System monitoring in the user interface .....	89
Home page.....	89
System management .....	90
Statistics monitor .....	90
Message Tracker .....	92
External monitoring of server status .....	93
Log file tracking .....	94
Event log .....	94
System logs .....	94
System statistics logs .....	95
User Interface logs .....	96
HTTP server logs.....	96
Other logs.....	96
Real-time viewing of log files .....	96
Set up tail as a Windows option .....	97
Troubleshooting with log4j file .....	98
Send log files to technical support .....	103
<b>Chapter 9. Events system</b>	<b>105</b>
Event levels .....	105
The events.xml file .....	106
EventRouters section of events.xml .....	107
MetadataProcessors section of events.xml .....	108
EventFilters section of events.xml .....	109
Log file event router .....	112
JMS event router.....	114
Oracle AQ event router.....	114
XML for JMS and AQ event routers.....	114
Persistence event router.....	120
The alerts.xml file .....	121
Send events by e-mail.....	121
Editing alerts.xml .....	123
Meta-data for e-mail events.....	125
Event tables .....	126
<b>Chapter 10. Getting started with Activator</b>	<b>147</b>
How the trading engine processes .....	147
Major components .....	148
Trading engine glossary .....	149

---

Configuration outline .....	151
Interoperability .....	153
Security guidelines .....	153
Maintenance tips .....	154
<b>Chapter 11. Trading configuration</b>	<b>155</b>
How profiles work .....	155
The community profile .....	156
Anatomy of a community profile.....	157
Add a community .....	160
Check list for community configuration.....	160
The partner profile .....	162
Anatomy of a partner profile.....	163
Add a partner.....	164
Searching for partners .....	165
Routing IDs .....	165
Exporting and importing profiles .....	166
Export a community profile.....	167
Export a partner profile .....	167
Import a partner profile.....	168
Import company profile as community profile .....	169
Automatic profile imports .....	169
Creating profiles outside the application.....	170
Firewalls and proxy servers .....	171
Activator in a firewall environment .....	172
Editing URLs to allow for firewalls.....	173
Getting a partner's external connection details.....	174
Proxy servers .....	175
Self-registration of AS1, AS2 partners .....	175
Wizard preparation.....	176
Using the partner wizard .....	177
Partner data form .....	178
Trading engine .....	178
Identity.....	178
Protocol .....	179
Transport details.....	179
Preferences .....	181
Security details .....	181
Binary and XML documents .....	182
Partner categories .....	182
<b>Chapter 12. Embedded transport servers</b>	<b>183</b>
Global embedded HTTP server .....	184
Global embedded SMTP server .....	185
Global embedded web services API server .....	186
Change community embedded server .....	187
Community embedded HTTP fields.....	188
Community embedded FTP fields.....	190

---

Community embedded SMTP fields .....	192
Embedded HTTP use cases .....	194
Case A .....	194
Case B .....	195
Case C .....	196
Case D .....	196
Case E .....	197
Case F .....	198
<b>Chapter 13. Delivery exchanges</b>	<b>201</b>
Four exchanges required .....	202
Trading delivery exchanges .....	203
Integration exchanges .....	206
Multiple integration pickup exchanges .....	208
Multiple integration delivery exchanges .....	208
Simulate sending messages to integration.....	210
Adding transports .....	211
Delivery exchange wizard .....	212
Open wizard on community summary page .....	212
Open wizard on partner summary page.....	213
Detached e-mail for community .....	213
Embedded e-mail for community .....	215
E-mail transport for partners .....	216
Embedded HTTP transport .....	219
Staged HTTP transport .....	221
HTTP transport for partners .....	221
File system transport .....	223
External FTP transport .....	226
Embedded FTP transport .....	230
Types of embedded FTP servers .....	231
FTP user accounts.....	232
Firewalls, load balancers and embedded FTP .....	235
Embedded FTP fields .....	236
SFTP transport .....	240
SFTP fields.....	241
Testing SFTP .....	244
Troubleshooting SFTP .....	245
JMS transport .....	245
Most providers .....	247
Oracle AQ.....	248
JMS fields .....	249
Testing JMS.....	252
IBM MQSeries transport .....	253
Web services API integration .....	255
Integration pickup .....	255
Integration delivery .....	256
Web services message protocol .....	257
Payload packaging .....	258

---

Payload unpackaging .....	258
Message meta-data .....	259
WS-Security handler.....	260
WS-Addressing handler.....	260
Default configuration.....	260
Message meta-data documents.....	261
Post-processing configuration details .....	265
Post-processing message meta-data.....	266
Adding post-processing elements .....	267
Methods for writing scripts .....	267
Post-processing events.....	268
Post-processing points to consider.....	269
Manage file system integration .....	269
Set up default file system integration.....	269
Use file system to set meta-data values .....	273
Post-processing to route inbound messages .....	273
Enable, disable, test exchanges .....	274
Set delivery exchange preferences .....	276
<b>Chapter 14. Staged HTTP</b>	<b>277</b>
Overview of staged HTTP .....	277
Staged HTTP configuration outline .....	278
Staged HTTP files to deploy .....	279
Deploy the web servlet .....	279
Deploy on Apache Tomcat.....	282
Deploy on WebLogic.....	282
Log on to servlet UI .....	283
Managing mailboxes .....	284
Add a mailbox .....	284
Partner connection to staged HTTP .....	286
Edit, delete, view mailboxes .....	287
Global settings.....	288
Staged HTTP UI fields .....	288
Message protocols for staged HTTP .....	289
File forwarding to bypass polling .....	291
High availability staged HTTP .....	292
<b>Chapter 15. Transport maintenance</b>	<b>295</b>
Opening maintenance pages .....	296
SMTP maintenance .....	296
HTTP maintenance .....	305
Staged HTTP maintenance .....	313
External FTP maintenance .....	316
Embedded FTP maintenance .....	325
Managing users of embedded FTP .....	329
Programmatically submit via embedded FTP .....	330
SFTP maintenance .....	331
File system maintenance .....	338

---

JMS maintenance .....	342
MQSeries maintenance .....	349
Web services integration maintenance .....	354
Web services trading maintenance .....	358
Proxy tab .....	362
From address and To address tabs .....	362
Message attributes tab .....	364
EDI Splitter tab .....	370
Inline processing tab .....	371
Schedule tab .....	371
Delivery criteria tab .....	373
Change embedded transports .....	375
<b>Chapter 16. Outbound HTTP proxy</b>	<b>377</b>
<b>Chapter 17. Certificates and keys</b>	<b>379</b>
PKI description .....	379
PKI options .....	380
The role of trust in PKI.....	381
Scalability.....	382
Certificate revocation.....	382
Dual-key pairs .....	382
Why use encryption and digital signatures .....	383
The trading engine encryption method .....	384
Encryption and signing summary .....	385
Ensuring data integrity and trust .....	387
Certificate basics .....	388
SSL authentication .....	388
Giving certificates to partners .....	390
Certificate exchange with Axway partners.....	391
Certificate exchange with other partners.....	391
Self-signed or CA certificates .....	391
When to get certificates .....	391
What to do with expiring certificates .....	392
Trusted roots .....	394
Auto importing of intermediate and root certificates .....	395
<b>Chapter 18. Manage certificates</b>	<b>397</b>
Certificates pages .....	397
Community and partner certificate pages .....	398
View certificate information .....	401
Certificate field descriptions .....	401
Add a certificate .....	406
Set up certificates for a community .....	407
Generate self-signed certificates .....	408
Import Entrust certificate .....	411
Import RSA Keon certificate .....	415
Import key pair in certificate file .....	418

---

Import certificates for partners .....	421
Export a certificate to a file .....	424
Delete certificate .....	427
Certificate exchange messaging .....	428
Types of CEM messages.....	428
Types of certificates in CEM requests.....	430
Send a CEM request .....	431
Track CEM requests .....	432
Certificate revocation lists .....	437
How CRL checking works .....	438
Obtaining CRL access information.....	439
Add a CRL .....	440
Manage CRLs .....	441
Advanced CRL settings.....	442
Analyze certificates for errors .....	445
<b>Chapter 19. Collaboration settings</b>	<b>447</b>
Hierarchy of outbound settings .....	447
View settings between partners .....	448
View or change default settings .....	450
Default collaboration settings .....	451
AS1 default settings.....	451
AS2 default settings .....	454
AS3 default settings .....	459
Secure file default settings.....	463
Generic e-mail default settings .....	466
RosettaNet 1.1 default settings .....	467
RosettaNet 2.0 default settings.....	468
Web services default settings.....	470
Reliable messaging default settings .....	473
Community collaboration settings .....	474
Partner collaboration settings .....	477
<b>Chapter 20. Inbound message validation</b>	<b>481</b>
Opening the validation page .....	481
Duplicate EDI documents .....	481
Signed or unsigned messages .....	483
Encrypted or plain text messages .....	484
Duplicate CSOS orders .....	485
<b>Chapter 21. Message handling</b>	<b>487</b>
Setting up message actions .....	487
Define message attributes .....	491
Define message actions .....	492
<b>Chapter 22. Message Tracker</b>	<b>495</b>
Message Tracker search controls .....	495

---

Message Tracker page .....	495
Custom search.....	496
Trading information.....	497
Date .....	501
Columns.....	501
Search results .....	501
Message details .....	503
Icons on search results page .....	505
Message receipts .....	505
Delete, resend, reprocess options .....	506
Sharing saved searches .....	508
Changing search default settings .....	508
Configure payload view .....	511
Forcing a document type for XML .....	513
<b>Chapter 23. Data backups and deletes</b>	<b>515</b>
Global backup configuration .....	516
Delete unwanted files and records .....	518
Purge all records, backup files .....	520
Configure event purging .....	521
<b>Chapter 24. FTP client scripting interface</b>	<b>523</b>
Levels of scripting .....	523
Editing the command set document .....	524
FTP tester tool .....	525
<b>Chapter 25. Test trading</b>	<b>529</b>
Configure for test trading .....	529
Start the test .....	531
Troubleshooting for e-mail testing .....	532
Complete the testing .....	533
<b>Chapter 26. Document Generator</b>	<b>535</b>
Create EDI or XML test documents .....	535
Run from a command line .....	538
Command line parameters .....	538
Command line format .....	539
<b>Chapter 27. ebXML support</b>	<b>541</b>
ebXML message lifecycle .....	542
Outbound ebXML processing .....	542
Inbound ebXML processing .....	543
ebXML message meta-data .....	544
ebXML meta-data descriptions.....	545
Required, optional meta-data for integration .....	550
Message meta-data documents .....	551
MMD example.....	551

---

Using an MMD to ping a partner .....	551
Using an MMD for a status request .....	554
Supported trading transports .....	554
Supported integration transports .....	555
Integration pickup options .....	555
Delivery integration options .....	558
ebXML message compression .....	558
Setting up a community for ebXML trading .....	559
Routing ID for ebXML .....	559
The community profile and CPAs .....	560
Sending ebXML via an intermediary .....	560
Overview of intermediary re-routing .....	561
Intermediary message-handling notes .....	562
Supported transports .....	563
Configuration of intermediary .....	563
Configuration of end points .....	564
Intermediary trading with HTTPS .....	564
Extracting KeyInfo element for a CPA .....	565
Managing CPAs .....	565
Importing a CPA .....	569
Importing a CPA template .....	569
Tools for CPAs .....	570
ebxmlCpaSchematronValidator .....	570
ebxmlCpaSecurityGuard .....	571
ebxmlCpaValidator .....	572
mmdGenerator .....	573
STAR BODs with ebXML .....	574
HL7 payloads with ebXML .....	576
Self-registration of ebXML partners .....	577
ebXML hub procedure .....	578
ebXML partner procedure .....	580
ebXML troubleshooting .....	582
<b>Chapter 28. RosettaNet support</b>	<b>585</b>
RosettaNet overview .....	585
RosettaNet configuration outline .....	586
Add DTD-based PIP .....	588
Add schema-based PIP .....	588
Configuring pipdefinitions.xml file .....	589
RNIF meta-data elements .....	596
Message meta-data documents .....	601
Special handling of meta-data .....	602
With MMDs .....	603
Without MMDs .....	603
<b>Chapter 29. Synchrony CSOS</b>	<b>607</b>
Overview of CSOS functionality .....	607
How it works .....	608

---

CSOS configuration for sending .....	609
CSOS configuration for receiving .....	610
Import CSOS signing certificate .....	611
CSOS certificate revocation lists .....	612
Identify CSOS purchase orders .....	612
Order identification tab .....	613
Order sources tab .....	620
Related documents tab.....	622
CSOS duplicate orders tab.....	624
Sign pending orders .....	625
EDI to XML conversion example .....	628
<b>Chapter 30. Transport guidelines</b>	<b>637</b>
Secure HTTP solutions .....	637
Embedded HTTP server.....	637
External staged HTTP server.....	640
Troubleshooting HTTP connections .....	641
Trading large messages .....	643
Disk space .....	644
Databases, firewalls and large messages .....	644
Considerations by transport .....	645
<b>Chapter 31. Synchrony integration</b>	<b>649</b>
Integrate with Synchrony Sentinel .....	649
Steps to integrate with Sentinel.....	650
Event meta-data sent to Sentinel .....	651
Message states reported to Sentinel .....	658
Integrate with Synchrony Integrator .....	661
Message picked up from Integrator .....	661
Message sent to Integrator.....	662
Integrate with Synchrony Gateway .....	663
Synchrony Gateway configuration .....	664
Activator configuration .....	666
<b>Index .....</b>	<b>669</b>



# User comments

Please tell us what we can do to make this documentation more complete and easier to read and use.

Send your comments to [documentation@us.axway.com](mailto:documentation@us.axway.com).



---

# 1

# System requirements

The following topics provide the hardware and software requirements for running Activator.

In addition, we strongly recommend reading the release notes for supplemental information about system requirements and installation. For release notes, go to the support web site at <http://support.cyclonecommerce.com>.

## Concepts

- ◆ [Hardware](#)
- ◆ [Software on page 3](#)
- ◆ [Port assignments and possible conflicts on page 9](#)
- ◆ [Security considerations on page 11](#)
- ◆ [About the user license file on page 12](#)
- ◆ [Technical support on page 12](#)

## 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 Activator software
- ◆ 200-400 MB for data storage, but possibly more (see [Temp directory requirement](#))
- ◆ SVGA monitor
- ◆ DVD 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.

## Changing the temp directory

By default Activator uses your operating system's temp directory. For instance, for UNIX and Linux this is /tmp. After installing, you can change the temp directory by editing the filereg.xml file in [install directory]\[build number]\conf.

To change the temp directory, open filereg.xml for editing and find the following property, which is commented out by default:

```
File name="temp" path=""
```

Uncomment the property and type the path for the new temp directory. Save the file and restart Activator server for the change to become effective.

## 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***

Activator supports the following Windows operating systems.

- ◆ Windows XP
- ◆ Windows Server 2003

You must have administrator rights to successfully install Activator.

If you plan to run Activator as a Windows service, we highly recommend creating a user account solely for starting the service upon log-on for Activator and related applications (for example, aXML Agent). This user account should be granted all Windows administrator rights. The account should be fully dedicated to 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 Activator or related application as a Windows service, open the properties for the GatewayInterchangeService 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 Activator server 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 Activator is started manually, make sure the dedicated user never logs out to ensure the application will run continuously.

When running as a service, manual configuration is required if you want the service to restart Activator server automatically if the server shuts down as the result of some failure. To configure automatic restarts, open the properties for the GatewayInterchangeService. On the Recovery tab, select **Restart the Service** in the fields for First, Second and Subsequent failures. Do not change the zero value of the Reset fail count after field. Click **OK**.

## UNIX

Activator supports the following UNIX operating systems.

- ◆ AIX 5.3
- ◆ HP-UX 11i
- ◆ HP-UX 11i v2 (PA-RISC)
- ◆ HP-UX 11i (IA-64) Itanium
- ◆ Red Hat Enterprise Linux 3 and 4
- ◆ Solaris 9 (SPARC)
- ◆ Solaris 10 (SPARC and x86)
- ◆ SUSE Linux Enterprise Server 10

### 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.

### Recommendation for Solaris users

If you use Solaris, we recommend raising the ulimit value to 8000. This configures the operating system to allow Activator to have more files open simultaneously, reducing the possibility of a processing bottleneck.

### 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 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.

Parameter	New value	Description
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.

## OS patches and JRE

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 operating systems.

This version of Activator supports Sun Java Runtime Environment 5. Before installing the application, check whether your OS supports JRE 5. If you do not have the required patches, do not install Activator. Install any required OS patches before installing Activator.

The following are guidelines for checking your OS.

### Linux, Solaris or Windows

Check the OS levels supported by the Sun Java 5 JRE at <http://java.sun.com/j2se/1.5.0/system-configurations.html>.

### AIX

Check the OS levels supported by the IBM Java 5 JRE at <http://www-128.ibm.com/developerworks/java/jdk/aix/service.html>. You can determine your patch level by running **oslevel -r**.

### HP-UX

Check the OS levels supported by the HP Java 5 JRE at <http://h18012.www1.hp.com/java/patches/index.html>. You can determine your patch level by running **uname -r**.

## Database

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.

The recommended maximum capacity of the Derby database, as used with Activator, is no more than 25,000 acknowledged messages or 50,000 unacknowledged messages. We recommend setting up a database purge routine to keep the number of records below the recommended maximums.

For instance, Activator can handle a traffic volume of up to 5,000 documents a day, an average of 3.5 documents per minute. This is the upper limit of the system's capacity. If Activator processes 5,000 acknowledged messages a day, the system should be configured to purge messages older than 5 days. If Activator processes 500 acknowledged messages a day, the system could be set to purge records older than 50 days. For information about purging, see [Delete unwanted files and records](#) on page 518.

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 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.

### **Warning about pop-up blocking, plug-ins, add-ons**

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

Moreover, some browser plug-ins or add-ons designed to enhance web server performance may interfere with the user interface. If you encounter UI pages that are slow to refresh, disabling plug-ins or add-ons is recommended.

In particular, if you use Internet Explorer do not install the Yahoo toolbar in your browser, as this has been demonstrated to interfere with the user interface.

## **Online help troubleshooting**

If you experience display problems while using the online help, clearing the browser's cache is recommended. Display issues may include the wrong page or no page displays for context-sensitive help or table of contents links do not match pages.

- 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**).

## **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 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 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 Activator server. For details see [UI usage with proxy servers](#) on page 73.

## Adobe Acrobat

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

# Port assignments and possible conflicts

Some transports and database types have default port assignments in 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 Activator to eliminate possible port conflicts or after installing if you encounter a problem.

The following table lists the default port assignments in 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
Statistics monitor	3579	This port, which is used by the trading engine statistics monitor, can be changed by adding a property to the tuning.properties file. To change the port, add the following property on a new line by itself in the file:  statsServer.port=[number]  Save the file and restart the server.  This property only must be added if the port is other than 3579.
		Tuning.properties is at [install directory]\[build number]\conf.
Embedded FTP (trading)	4021	This port, which the trading engine uses for an embedded server, can be changed when adding an embedded FTP server.

**Table 1 - Port assignments**

Description	Default port	Where to change it
Embedded SMTP	4025	This port, which the trading engine uses for an embedded server, can be changed by clicking <b>Configure the global embedded SMTP server</b> on the main trading configuration page in the user interface.
Embedded HTTP or HTTPS	4080	This port, which the trading engine uses for an embedded server, can be changed by clicking <b>Configure the global embedded HTTP server</b> on the main trading configuration page in the user interface.
Embedded FTP (integration)	5021	This port, which the trading engine uses for an embedded server, can be changed when adding an embedded FTP server.
Embedded web services API server	5080	This port, which the trading engine uses for an embedded server, can be changed by clicking <b>Configure the global embedded web services API server</b> on the main trading configuration page in the user interface.
Used by nodes to communicate with the executive	5107	..\bin\startServer.cmd
User interface HTTP	6080	This port is used to access the user interface in a browser via HTTP. This can be changed by going to the system management page and clicking <b>Configure UI connection</b> .
User interface HTTPS	6443	This port is used to access the user interface in a browser via HTTPS. This can be changed by going to the system management page and clicking <b>Configure UI connection</b> .

**Table 1 - Port assignments**

Description	Default port	Where to change it
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 Activator. Alternately, you can configure 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 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 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 Activator.
- 4** Control access to the computer running Activator to authorized users.
- 5** If you use an external database on a different computer than the one running 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.

## Technical support

If you have problems with this software, try to find solutions in the user documentation and the readme file. If you need help, contact technical support at [support2@us.axway.com](mailto:support2@us.axway.com).

### Reporting an issue

Before contacting support, be prepared to provide a description of the problem. This includes symptoms and the sequence of events leading to the problem. Also, be prepared to provide the following information for more efficient service:

- The version and build you are using
- The platform on which you are running the application

You can find the version number in **Help > About** in the user interface. Click **Details** for the build number.

### Support web site

Users who have purchased support can obtain help 24 hours a day for identifying and resolving problems from the support web site at <http://support.cyclonecommerce.com/>. The support area has answers to

frequently asked questions, technical papers and the latest versions of software you can download. A password is required for using the support area.



## 1. System requirements

---

# 2 Installing and starting the server

The following topics discuss installing Activator server. Also covered is starting the server the first time and logging on to the user interface in a browser.

Important information about installation is provided here, but details about the installer are available. In the installer directory tree, go to the Documentation folder and open **index.htm** in a browser. Or, after launching the installer, click the **Documentation** link on the welcome page.

You can install Activator on computers that meet the hardware and software criteria described in [System requirements](#) on page 1.

## Concepts

- ◆ [Installation outline](#)
- ◆ [Before installing](#) on page 17
- ◆ [Guidelines for installing on UNIX](#) on page 17

## Procedure

- ◆ [Start the server on Windows](#) on page 18
- ◆ [Start the server on UNIX](#) on page 19
- ◆ [Run as a Windows service](#) on page 19
- ◆ [Open the user interface](#) on page 22
- ◆ [Configuring external SMTP server](#) on page 23
- ◆ [Restarting for server performance](#) on page 23
- ◆ [Stop the server](#) on page 24
- ◆ [Uninstalling](#) on page 24

## Installation outline

The following are the basic steps in setting up Activator.

- 1 Review [Before installing](#) on page 17 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 39 before proceeding.

**3** Install Activator.

For details about the installer, go to the Documentation folder in the installer directory tree and open **index.htm** in a browser. Or, after launching the installer, click the **Documentation** link on the welcome page.

If installing on UNIX, see [Guidelines for installing on UNIX](#).

The installer lets you choose whether to migrate company and partner profile data from Interchange or Activator 4.2.x. If you choose this option, the installer exports profiles from the earlier Interchange or Activator and writes the files to [build number]\profiles\staged of the new application's installation directory tree. See [Import all profiles](#) on page 41 for how to import the profiles to the trading engine.

- 4** Start the server. This creates the database tables for the embedded Derby database. See [Start the server on Windows](#) on page 18 or [Start the server on UNIX](#) on page 19.
- 5** Open the user interface in a browser. See [Open the user interface](#) on page 22. If this is the first time you are logging 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 with Activator](#) on page 147.
- 9** While configuring the application you may have to edit system files or add your own custom scripts or code. This is normal 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 keeping copies of all files 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 56.

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 36.

## Before installing

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

- ◆ Make sure you know the location of the license.xml file. The installer prompts you for it. You cannot install without the license file. See [About the user license file](#) on page 12. 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.

## Guidelines for installing on UNIX

When installing on UNIX, the home directory for Activator must not be automounted or on an automounted file system. 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. Activator cannot be installed on automounted volumes because of automatic unmounting of such drives.

If you use NFS, we recommend putting on an NFS mount only files that must be shared. These include:

- ◆ Backup files
- ◆ Key store (PSE) files
- ◆ CRL files
- ◆ File system pickup and delivery directories

Do not share:

- ◆ Log files
- ◆ Virtual data temporary (overflow) files
- ◆ Any other temporary files

When exporting a directory on the NFS server, use the **sync** option to make sure all writes are sent to disk before the NFS call returns to the client. This may degrade performance somewhat, but ensures consistency of the file system.

Hard mounts are strongly recommended for NFS. File systems that are mounted read-write or that have executable files should be mounted with the hard option.

When hard mounting is used, all operations on the NFS mount will block until the operation is completed. If the NFS server is down, the client will block and retry continuously until the server restarts and completes the operation. When using hard mounts, use the **bg** and **intr** options to background the mount if it is not initially successfully and to allow a process that is waiting on an NFS operation to complete to be interrupted.

If you must use soft mounts, do not put read-write file systems or file systems containing executables (like JAR files) on soft mounts. If you have no choice other than to use soft mounts, Activator may encounter input/output errors. The writing of message backup files has been made tolerant of I/O errors, but there are other parts of the application that may not behave well when encountering an I/O error.

**Note:** If you use AIX, read this note. 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, Activator can be installed into a directory on that mount point just as it can with a non-LVM volume. Activator software itself is not aware of whether or not this is an LVM mount point.

## 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 > Axway > 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 axway 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 Activator server as a Windows service. This causes the application to start or stop with the computer. When 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 Activator as a Windows service, we highly recommend creating a user account solely for starting the service upon log-on for Activator and related applications (for example, aXML Agent). This user account should be granted all Windows administrator rights. The account should be fully dedicated to 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 Activator or related application as a Windows service, open the properties for the GatewayInterchangeService 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 Activator server 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 Activator is started manually, make sure the dedicated user never logs out to ensure the application will run continuously.

When running as a service, manual configuration is required if you want the service to restart Activator server automatically if the server shuts down as the result of some failure. To configure automatic restarts, open the properties for the GatewayInterchangeService. On the Recovery tab, select **Restart the Service** in the fields for First, Second and Subsequent failures. Do not change the zero value of the Reset fail count after field. Click **OK**.

## ***Install service***

Use this procedure if you are not already running the server as a Windows service and want to.

If you upgraded recently and did not install the Windows service during installation, but ran the older version as a Windows service, you must stop the server and uninstall the service before installing the new service for the upgraded application. See [Uninstall service](#) on page 21.

### **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: **GatewayInterchangeService -i**

Domain account: **GatewayInterchangeService -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***

Use this procedure to stop running the server as a Windows service

### **Steps**

- 1** Open a command prompt window.
- 2** Change the directory to [install directory]\[build number]\bin.
- 3** Run the command **GatewayInterchangeService -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 Activator server. This performs the same action as described in [Install service](#) on page 20, 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 **GatewayInterchangeService**.

## 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 18 or [Start the server on UNIX](#) on page 19.
- ◆ 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 > Axway > 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/**
- 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.

**Host** is the fully qualified domain name or IP address of the computer running Activator server.

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 78..

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.

## Restarting for server performance

A best practice to maintain healthy performance of Activator is to restart the server on a regularly scheduled basis, at least once a month. This requires you to briefly suspend trading and tell partners when your trading engine is being restarted or shut down for maintenance.

The recommendation for restarting the server stems from the nature of a software application written in Java. It is not due to a fault within Activator.

If you do not institute a restart schedule, you must, in the least case, restart the server when Activator imports new trusted root certificates. The Java virtual machine (JVM) processing nodes are responsible for trusted roots, not Activator. Restarting flushes the operational cache of expired certificate data.

The following is the correct way to restart:

Stop the server on the computer running Activator. If you operate the trading engine in a cluster of multiple computers, stop the servers on all computers. All nodes in the cluster must be stopped. Restart only after all servers have been stopped. Serially restarting nodes does not achieve a full

stop. All trading activity must be suspended while Activator is shut down. Wait at least one minute before restarting. You also can restart the computer, as this also flushes memory buffers.

If you run Activator as a Windows service, stop the service to stop the server gracefully. If you do not reboot the computer, restart the service to start the server again.

## 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

Stop the server, launch the uninstaller and follow the prompts. You can uninstall in GUI or console mode. Use the setup file or path for your operating system.

### GUI mode

Windows    Select **Start > Axway Software > Synchrony [version number] > Uninstall**

UNIX        setupUNIX.sh UnInstall

### Console mode

Windows    setupWin32 UnInstall InstMode Console

UNIX        setupUNIX.sh UnInstall InstMode Console



# 3 Upgrading

This chapter provides information and guidance for upgrading to newer versions of Activator.

## Procedure

- [Directory tree's role in upgrading](#)
- [Upgrading from version 5.1 or later](#) on page 26

## Concepts

- [Change database from Sybase to Derby](#) on page 32
- [File comparison tools for upgrades](#) on page 36
- [Migrating version 4.2 trading profiles](#) on page 39

## Directory tree's role in upgrading

When you install Activator 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.

## Upgrading from version 5.1 or later

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

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

### ***Back up database***

If you follow the correct procedure, all data are preserved in the upgrade.

We highly recommend preserving the build number directory of the currently installed application, at least until you have installed and started the upgrade and are satisfied with performance.

The build number directory of the currently installed application contains the files for the embedded database. As long as you do not delete the directory, you have a back-up of the database.

Database tables for the embedded database are in [install directory]\[build number]\corelib\db\derby. When Activator is upgraded, data from the previous embedded database are copied to a new embedded database.

### **Database change from Sybase to Derby**

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 [Change database from Sybase to Derby](#) on page 32.

If you are unsure of your current database, see [Database configuration tool](#) on page 52 to look up database information.

## ***Back up common directory***

Backing up the application's common directory also is recommended before upgrading. The common directory contains back-up files, data files and certificate key files. After upgrading, the new application uses and adds to the common directory. Backing up before upgrading provides insurance if you want to revert to the previous version after upgrading.

You can copy the common directory to some other directory on your file system. Or you can copy and rename the directory, leaving the renamed copy in the installation directory.

## ***Carry functionality forward***

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.

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 56.

If you edited the filereg.xml file in the conf directory and placed files you registered in filereg.xml in the application's conf directory, you must edit the new filereg.xml after upgrading and copy the files you added to the old conf directory to the new conf directory.

## **Note for ebXML users**

If you trade messages via the ebXML message protocol and are upgrading from a version before 5.4.2, a change in the way the database stores ebXML messages may affect you.

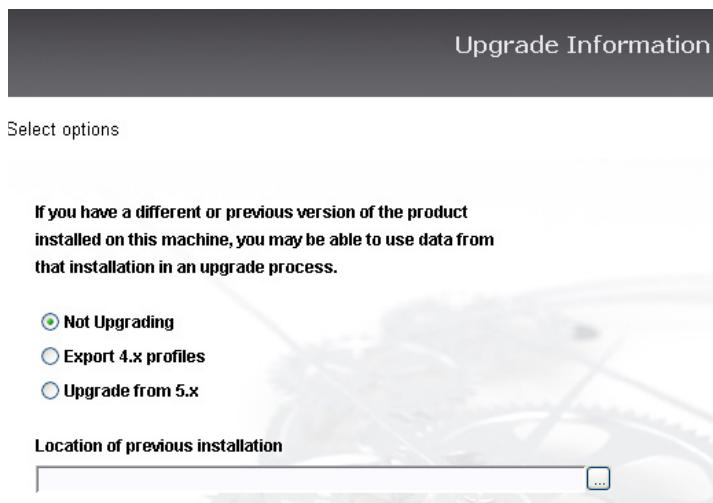
When you start the server the first time after upgrading, the application will reorganize ebXML data in the database. Depending on the number of ebXML database records, the consolidation may result in a start-up delay of several minutes or more the first time the server is started. While the consolidation is occurring, status messages reporting the progress of the data conversion will display every 30 seconds on the console window and in the server log file.

If there are less than 100,000 ebXML messages in the database, the first-time server start-up may be delayed by 10 or 15 minutes. In rare cases where there are millions of ebXML database records, the delay could be hours long. In any event, the delay occurs only the first time the server is started after the upgrade.

## **Upgrade steps**

- 1** Review [System requirements](#) on page 1 to make sure your hardware and software meet requirements for the application upgrade.
- 2** Stop Activator server of the currently installed application.
- 3** Back up the application's common directory before upgrading.
- 4** Install the new version (see [Installing and starting the server](#) on page 15).

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.



**Figure 1. Installer upgrade information window**

The following describes 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.

### **Upgrade from 4.x**

Select **export 4.x profiles** when upgrading from Interchange or 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 39.

### **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.

---

- 5 If upgrading from Activator with a Sybase database to 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 6.

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

- 6 If upgrading from Activator with a Sybase database to Activator with a Derby database and you want to migrate data to the new Activator, see [Change database from Sybase to Derby](#) on page 32 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 4, 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 52 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

- 7 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 56.

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 36.

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.

- 8 If you exported database files in step 6 to change databases, use the data migration tool to import the data. See [Change database from Sybase to Derby](#) on page 32.
- 9 Start the server.
- 10 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).

- 11 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.

## Change database from Sybase to Derby

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 52.

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 Gateway 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 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 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 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, 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 in the bin directory named axwayInstallation.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. Due to length, only a small portion of the file is shown.

```
\ 1172685014936 0
\Interchange-ActivatorREADME.txt 1168647483000 386
```

```
\JettyLicense.pdf 1172639319000 20341
\TransactionDirectorREADME.txt 1168647506000 385
\bin\ 1172685035806 0
\bin\GatewayInterchangeService.exe 1172623858000 106496
\bin\GatewayInterchangeService.ini 1172685034034 1563
\bin\GatewayInterchangeService.ini.bak 1172685033994 1596
\bin\OBOE.properties 1088720207000 566
\bin\PLORENTZ-T41_environment.cmd 1172685033964 835
\bin\admin$2.ico 1172685030679 2238
\bin\admin.cmd 1110926070000 157
\bin\dbConfig$1.ico 1172685030258 2238
\bin\dbConfig.cmd 1109774743000 1144
\bin\docGen$3.ico 1172685030699 2238
\bin\docGen.exe 1172685029978 114688
\bin\docGen.lax 1172685030048 3493
\bin\environment.cmd 1172685022938 1852
\bin\lax.jar 1172685029928 41642
\bin\manageNode.cmd 1147214638000 574
\bin\netConfig.cmd 1172685022958 765
\bin\nodeInfo.cmd 1147214439000 576
\bin\startServer.cmd 1149089582000 3092
\bin\startServer.ico 1172685030138 2238
\bin\systemuser.cmd 1128982387000 1915
\bin\tail.exe 1088720097000 16896
\bin\unlockUser.cmd 1166723839000 762
```

## 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. When you use the -output paramemter, the tool writes a text file in the bin directory. For example, the following command writes a file name upgradediff.txt to the bin directory:

### **upgradediff -output upgradediff.txt**

The following is a sample of the content. Records with the prefix <Diff> are the files detected to be of different sizes.

```
<Both> \bin\ bin is a folder that exists both in
C:\install_directory\b2299\bin and C:\install_directory\b2328\bin
<Same> \bin\GatewayInterchangeService.exe
GatewayInterchangeService.exe is a file that exists both in
C:\install_directory\b2299\bin\GatewayInterchangeService.exe and
C:\install_directory\b2328\bin\GatewayInterchangeService.exe
```

```
<Same> \bin\GatewayInterchangeService.ini
GatewayInterchangeService.ini is a file that exists both in
C:\install_directory\b2299\bin\GatewayInterchangeService.ini and
C:\install_directory\b2328\bin\GatewayInterchangeService.ini
<Same> \bin\GatewayInterchangeService.ini.bak
GatewayInterchangeService.ini.bak is a file that exists both in
C:\install_directory\b2299\bin\GatewayInterchangeService.ini.bak and
C:\install_directory\b2328\bin\GatewayInterchangeService.ini.bak
<Same> \bin\OBOE.properties OBOE.properties is a file that exists both
in C:\install_directory\b2299\bin\OBOE.properties and
C:\install_directory\b2328\bin\OBOE.properties
<Same> \bin\PLORENTZ-T41_environment.cmd PLORENTZ-T41_environment.cmd
is a file that exists both in C:\install_directory\b2299\bin\PLORENTZ-
T41_environment.cmd and C:\install_directory\b2328\bin\PLORENTZ-
T41_environment.cmd
<Same> \bin\admin$2.ico admin$2.ico is a file that exists both in
C:\install_directory\b2299\bin\admin$2.ico and
C:\install_directory\b2328\bin\admin$2.ico
<Same> \bin\admin.cmd admin.cmd is a file that exists both in
C:\install_directory\b2299\bin\admin.cmd and
C:\install_directory\b2328\bin\admin.cmd
<Only> \bin\admin.url admin.url is a file that exists only in previous
location C:\install_directory\b2299\bin\admin.url
<Diff> \bin\axwayInstallation.txt axwayInstallation.txt is a file that
exists both in C:\install_directory\b2299\bin\axwayInstallation.txt
and C:\install_directory\b2328\bin\axwayInstallation.txt and has old
size of 585544 and new size of 585254
```

## 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 in the bin directory named cycloneComparison.txt. The following is a sample of the content.

```
<Diff-CurNewer> \ is a folder that exists in old
C:\install_directory\b2299\ and new C:\install_directory\b2328\ and
has old time of Feb.14.07 10.26.56.995 and new time of Feb.28.07
10.50.14.936
```

```
<Same> \Interchange-ActivatorREADME.txt Interchange-
ActivatorREADME.txt is a file that exists in old
C:\install_directory\b2299\Interchange-ActivatorREADME.txt and new
C:\install_directory\b2328\Interchange-ActivatorREADME.txt
<Diff-CurNewer> \JettyLicense.pdf JettyLicense.pdf is a file that
exists in old C:\install_directory\b2299\JettyLicense.pdf and new
C:\install_directory\b2328\JettyLicense.pdf and has old time of
Feb.13.07 22.19.06.000 and new time of Feb.27.07 22.08.39.000
<Same> \TransactionDirectorREADME.txt TransactionDirectorREADME.txt is
a file that exists in old
C:\install_directory\b2299\TransactionDirectorREADME.txt and new
C:\install_directory\b2328\TransactionDirectorREADME.txt
<Diff-CurNewer> \bin\ is a folder that exists in old
C:\install_directory\b2299\bin\ and new
C:\install_directory\b2328\bin\ and has old time of Feb.14.07
10.36.32.913 and new time of Mar.02.07 10.38.12.279
<Diff-CurNewer> \bin\GatewayInterchangeService.exe
GatewayInterchangeService.exe is a file that exists in old
C:\install_directory\b2299\bin\GatewayInterchangeService.exe and new
C:\install_directory\b2328\bin\GatewayInterchangeService.exe and has
old time of Feb.13.07 18.09.04.000 and new time of Feb.27.07
17.50.58.000
```

## Migrating version 4.2 trading profiles

If you are upgrading from Interchange 4.2.x to 5.0 or later, you can copy company and partner profiles to the new application. You cannot, however, migrate database 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 41 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 Interchange 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 Interchange 4.2.x bin directory, go to step 6.

- 1** Locate the import utility in the Interchange 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 Interchange 4.2.x to Activator 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 166.

**Note:** In 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 started, 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 Interchange 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 447 and [Inbound message validation](#) on page 481.

## 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 269 for more information.

If you change the target location from that used by Interchange 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 Interchange 4.1.x or earlier, those versions do not export all profile information that 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 45
- ◆ [Installation directory tree](#) on page 49
- ◆ [Single sign-on interface](#) on page 54
- ◆ [Planning for upgrades and disaster recovery](#) on page 56
- ◆ [Running the UI over HTTPS](#) on page 58

## Procedure

- ◆ [Database configuration tool](#) on page 52
- ◆ [Binding network interfaces](#) on page 53

## 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 some 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 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 <a href="#">Database configuration tool</a> on page 52.
environment	Sets environment variables that are specific to all nodes running on a particular computer. This tool is not for use by end users.
manageNode	Lets you add, remove, start, stop and restart nodes. Run <b>manageNode</b> to display instructions for using the tool and a list of available parameters. Only a user associated with the admin role can use this tool.
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 GatewayInterchangeService.ini file, as well as the <machine>_environment.cmd script that is used when the trading engine is run from a command prompt.
nodeInfo	Displays information about the nodes in a cluster, machines in a cluster and node status. Run <b>nodeInfo</b> to display instructions for using the tool and a list of available parameters. Only a user associated with the admin role can use this tool.
startServer	Starts Activator server application. See <a href="#">Start the server on Windows</a> on page 18 or <a href="#">Start the server on UNIX</a> on page 19.
stopServer	The UNIX command to stop Activator server application. See <a href="#">Stop the server</a> on page 24
systemuser	Changes or adds a system user and password. This user is only used by Activator system. This tool is not for use by end users, except in connection with SSO.

**Table 2 - Tools in the application's bin directory**

Tool	Description
unlockuser	Unlocks a user who is blocked from logging on to the user interface after exhausting the number of log on retries. See <a href="#">Unlocking a blocked user</a> on page 88.

## 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.
certScan	Scans public-key certificates and reports information, warnings and errors. See <a href="#">Analyze certificates for errors</a> on page 445.
dataMover	Migrates data from one database to another. This tool is to be used only by some Activator users under the supervision of technical support. See <a href="#">Change database from Sybase to Derby</a> on page 32.

**Table 3 - Tools in the application's tools directory**

Tool	Description
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.
DeleteUploadedDirectorDocs.sql	This script deletes all records in the 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.  For information about how to submit log files to technical support through the user interface, see <a href="#">Send log files to technical support</a> on page 103.
diff	The diff tool is similar to the Unix diff tool and the Windows comp tool. It improves upon these tools by reporting the offsets of differences even in binary files. Also, it is platform independent, and it sets exit codes to allow shell scripts or batch files to make use of the tool. Very large files are processed using Java nio buffers for efficiency. The tool provides help if you invoke it with -?.
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.
ebxmlCpaSchemaValidator	Performs tests on the content of the ebXML CPA. The tool makes sure matching elements in each PartyInfo element of the ebXML CPA are consistent. See <a href="#">Tools for CPAs</a> on page 570.
ebxmlCpaSecurityGuard	Used for digitally signing CPAs. Its various functions all relate to signing and verifying digital signatures of a CPA. See <a href="#">Tools for CPAs</a> on page 570.

**Table 3 - Tools in the application's tools directory**

Tool	Description
ebxmlCpaValidator	Performs a schema validation on a CPA. See <a href="#">Tools for CPAs</a> on page 570.
exportProfile	Exports community and partner profiles to XML files. Community profiles are exported as partner profiles. Partner profiles also can be exported as partner profiles, either singly or in a batch. Run exportProfile without parameters to display directions for using the tool. This tool is only for use with Activator 5.4 or later.
fillInMessage Direction	<p>For messages traded before upgrading to version 5.4 or later of Activator, this tool adds meta-data to the database regarding the direction of traded messages (inbound, outbound). Message direction displays in the search results of Message Tracker.</p> <p>Use of this tool is optional if you have used versions earlier than 5.4. The tool is not needed if you did not use a version earlier than 5.4.</p> <p>Run this tool only when Activator server is not running. If the database contains thousands of records, it may take hours for the tool to run. If you start the tool and end the process before it is completed, you can re-start the tool later and it will pick up where it left off.</p>
ftpTester	Verifies interoperability of the trading engine with FTP servers. See <a href="#">FTP tester tool</a> on page 525.
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 <a href="#">JMS transport</a> on page 245.
keyInfoWriter	Extracts KeyInfo element information from a certificate for use in a CPA for ebXML trading. See <a href="#">Extracting KeyInfo element for a CPA</a> on page 565.
listTimeZones	Lists all available time zones for the JRE in use on a computer.

**Table 3 - Tools in the application's tools directory**

<b>Tool</b>	<b>Description</b>
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 <a href="#">Purge all records, backup files</a> on page 520.
mmdGenerator	Used to generate all possible MMDs or a specific MMD for an ebXML CPA. See <a href="#">Tools for CPAs</a> on page 570.
netInfo	Finds network interfaces for a computer. See <a href="#">Binding network interfaces</a> on page 53.
partyInfo	Lists the names and details about the community, partner and WebTrader profiles configured in Activator and the totals for each profile type. This tool provides a way to obtain information about profiles outside of the user interface.
postInstall	Sets up Activator server as a Windows service and adds shortcuts on the Start menu. See <a href="#">Use postInstall to add Windows service</a> on page 21.
postInstall.dat	See postInstall.
rejectInprocess Messages	Sets 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 <a href="#">SFTP transport</a> on page 240.
sysInfo	Displays system information such as the operating system, memory statistics, JVM class path and JVM library path. This information also writes to [install directory]\[build number]\logs\sysInfo.log.
uiSslConfig	Helps in the configuration for running the user interface over HTTPS. See <a href="#">Running the UI over HTTPS</a> on page 58.

**Table 3 - Tools in the application's tools directory**

Tool	Description
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 <a href="#">File comparison tools for upgrades</a> on page 36.
upgradeDiff	Recursively compares the sizes of system files in the old and new installation directory trees. This tool is used when upgrading. See <a href="#">File comparison tools for upgrades</a> on page 36.
upgradeList	Generates a snapshot of the entire application installation directory tree. This tool is used when upgrading. See <a href="#">File comparison tools for upgrades</a> on page 36.

## Installation directory tree

Application files used by 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 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 Activator. This directory also stores database drivers.
doc	User documentation in PDF format for 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 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 Activator.
samples	Contains sample code for demonstrating uses of various optional features.
site	Recommended directory for users to document custom changes made to Activator and for storing custom scripts and code as an aid for upgrading or disaster recovery. See <a href="#">Planning for upgrades and disaster recovery</a> on page 56.
tools	Has scripts for executing various troubleshooting and configuration tools.
util	Contains files and user documentation for optional utilities.

**Table 4 - Build number directory descriptions**

Directory	Description
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.
cptemplates	Where Activator stores imported CPA templates used for ebXML trading.
data	Contains copies of files processed by Activator. Message Tracker uses these files. Users can control the volume of files through controls in the user interface.
diagnose	This directory is used by the diagnose log file packaging tool to store compressed files. See <a href="#">diagnose</a> on page 46.
filestore	Contains copies of the files uploaded to Director.
webtrader	Activator default directory for storing documents exchanged between a web trader and a community sponsor.

# Database configuration tool

Database driver and connection information for 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.



**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 Activator server.
- 2** 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**.
- 3** 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.
- 4** Restart 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 server IP binding**. 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.

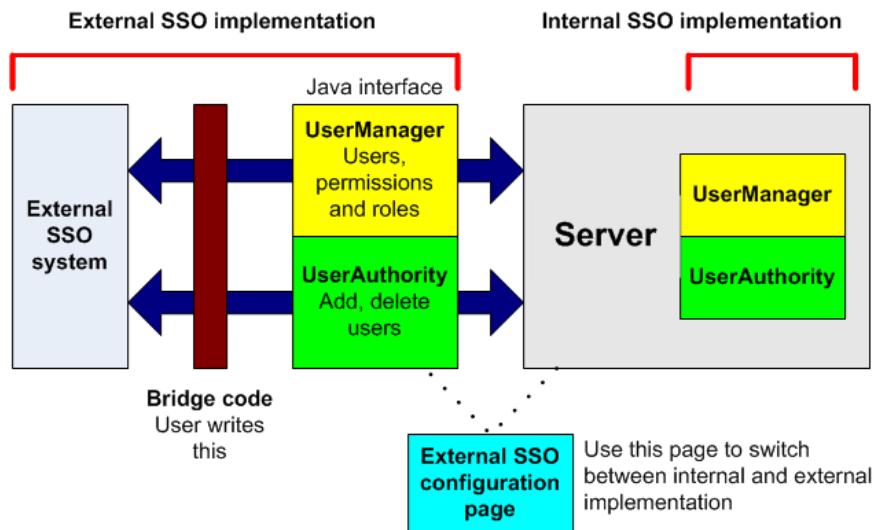
## Single sign-on interface

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 Activator server when a user is invalid.

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**

The external SSO configuration page referenced in Figure 4 is opened with the following URL in the user interface:

<http://host:6080/ui/sysmgmt/ExternalSSO>

Host is the fully qualified domain name or IP address of the computer running the server. Only users who configure an external SSO system should use this page.

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 Activator.

Managing 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.

Activator provides a file system directory to help in documenting custom changes. The directory is [install directory]\[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. Your source code should refer to this directory as ..\site\conf.

#### **doc**

Store text documents containing notes about custom changes. These can be notes about any changes that would be a useful reference for someone performing 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.

**jars**

Store Java archive (JAR) files and class files for in-line processors, pluggable transports, JMS, custom parsers. Any classes in this directory are included automatically in the classpath before [install directory]\[build number]\jars. This includes JAR files; there is no need to explicitly add them to the classpath.

# Running the UI over HTTPS

The default way for browsers to connect to the application server's user interface is via HTTP. Typically, the URL a browser uses to connect is `http://host:6080/ui`, where host is the name of the computer running the server. Optionally, you can have browsers connect instead via HTTPS (HTTP over SSL). You also can allow connections via HTTP and HTTPS at the same time. The following topics explain how to configure this.

## **Configure HTTPS**

Use this procedure to configure the server so browsers can log on to the user interface via HTTPS.

### **Steps**

- 1** Click **System management** on the toolbar to open the system management page.
- 2** Click **Configure UI connection** to open the Configure UI connection page.

## Configure UI connection

Select whether browsers can connect to the user interface via HTTP, HTTPS or both. If you select HTTPS, add a server certificate on the UI certificates tab. You can use a self-signed or CA certificate. The certificate secures the connection between browsers and the server. If you select HTTPS and require client authentication, add the client's trusted root certificate.

General | UI certificates | Trusted root certificates

Specify the port for HTTP or HTTPS or both

UI connections made via HTTP  
HTTP Port: \* 6080

UI connections made via HTTPS  
HTTPS Port: \* 6443

Require client authentication

In addition to clicking Save, you must restart the server or restart all nodes and the user interface for your changes to take effect.

**Save**

**Figure 5. UI configuration page for HTTP and HTTPS**

If you are opening the page the first time, connections via HTTP already is configured by default. You can leave the page as-is or add configuration for connecting via HTTPS. You cannot disable connections via HTTP until you have configured HTTPS. Once HTTPS has been configured, you can return to this page and select to have browsers connect via HTTP or HTTPS or both.

- 3 On the General tab, select **UI connections made via HTTPS**.

Although port 6443 is suggested, you can change the number as your situation requires.

- 4 Click **Save**.
- 5 Select the UI certificates tab and click **Add a certificate** to open the certificate wizard.

You can add a self-signed or a CA certificate. The certificate has a public-private key pair. The certificate is used to secure connections between browsers and the server.

If you choose to add a self-signed certificate, you can accept all default values in the certificate wizard.

The steps for adding a server certificate are the same as adding a certificate for a community profile. See [Add a certificate](#) on page 406 for more information.

After adding the certificate, the General tab displays again.

- 6** Select the UI certificates tab again. The certificate you added in step 5 is listed. You can click the certificate's name to display details.
- 7** Select the circle preceding the certificate's name and click **Save**.
- 8** On the General tab, check again that **UI connections made via HTTPS** is selected.
- 9** If are configuring HTTPS and selected **Require client authentication**, select the Trusted roots certificates tab and add a trusted root certificate.

With this option, the server requires the user's browser to send a certificate back to the HTTPS server. 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 file.

- 10** To complete the configuration you must do one of the following:
  - ♦ Restart the server

or

  - ♦ Restart all nodes and the user interface. Go to the system management page and click **Stop all nodes**. On the Stop all nodes page, click **Restart all nodes** and **Yes, include the user interface**. Click **Stop/restart**. Note that restarting the user interface ends your browser session.
- 11** Inform users of the URL needed to connect from a browser to the user interface. If you use the suggested port of 6443, the URL is:

`https://host:6443/ui`

where host is the fully qualified domain name or IP address of the computer running the server.

## **Switch between HTTP and HTTPS**

Once connections via HTTPS have been configured, you can return to the UI configuration page and select to allow browser connections via HTTP or HTTPS or both.

If you change the configuration, click **Save**. You also must do one of the following:

- ♦ Restart the server
- or
- ♦ Restart all nodes and the user interface. Go to the system management page and click **Stop all nodes**. On the Stop all nodes page, click **Restart all nodes** and **Yes, include the user interface**. Click **Stop/restart**. Note that restarting the user interface ends your browser session.





# 5 Navigating the user interface

The web-based user interface has been designed for ease of use. It has 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 to walk you through configuration tasks.

## Concepts

- [The toolbar](#)
- [Navigation aids for the trading engine](#) on page 65
- [Help for users of earlier versions](#) on page 65

## The toolbar

The primary navigation aid is the top toolbar. The options available on your toolbar depends on the functionality allowed by your user license.



**Figure 6. 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.
	CSOS	Configure and view orders for compliance with the Controlled Substance Ordering System. This displays only when users have a license for CSOS.

Icon	Name	Link description
	ePedigree	Configure the trading engine to process pedigree messages. This displays only when users have a license for ePedigree.
	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 JVM nodes and 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	Product information and user documentation.

# 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 7 shows the community navigation graphic, and Figure 8 shows the partner graphic. The graphics appear on the community or partner summary page and related pages.



**Figure 7. Community navigation graphic**



**Figure 8. Partner navigation graphic**

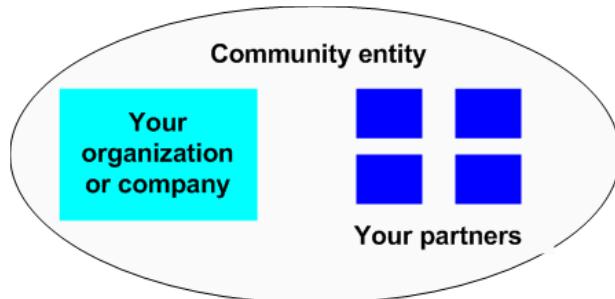
For more information see [Anatomy of a community profile](#) on page 157 and [Anatomy of a partner profile](#) on page 163.

## Help for users of earlier versions

If you have used 4.x versions, 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 will notice a change in the way the new application manages trading profiles. Where Interchange 4.x 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 9 illustrates this concept, which is reflected in the user interface. For more information about communities, see [Trading configuration](#) on page 155.



**Figure 9. The trading engine view of a community entity**

The following topics are to help users of earlier versions of Interchange 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:

Configuration for sending alert or notification messages by e-mail is performed by editing the alerts.xml file. See [The alerts.xml file](#) on page 121.

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 [Delivery exchange wizard](#) on page 212.

For information about delivery exchanges, see [Delivery exchanges](#) on page 201.

## 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 [Delivery exchange wizard](#) on page 212.

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 362

## 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 269.

## 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 265 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 487.

## 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 [Delivery exchange wizard](#) on page 212.

For information about delivery exchanges, see [Delivery exchanges](#) on page 201.

## 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 377 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 447.

## 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 269.





# 6 UI usage with proxy servers

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

## Concepts

- [Deployment in proxy environment](#)
- [Forward proxy](#) on page 73
- [Reverse proxy](#) on page 74

## Deployment in proxy environment

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.

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.

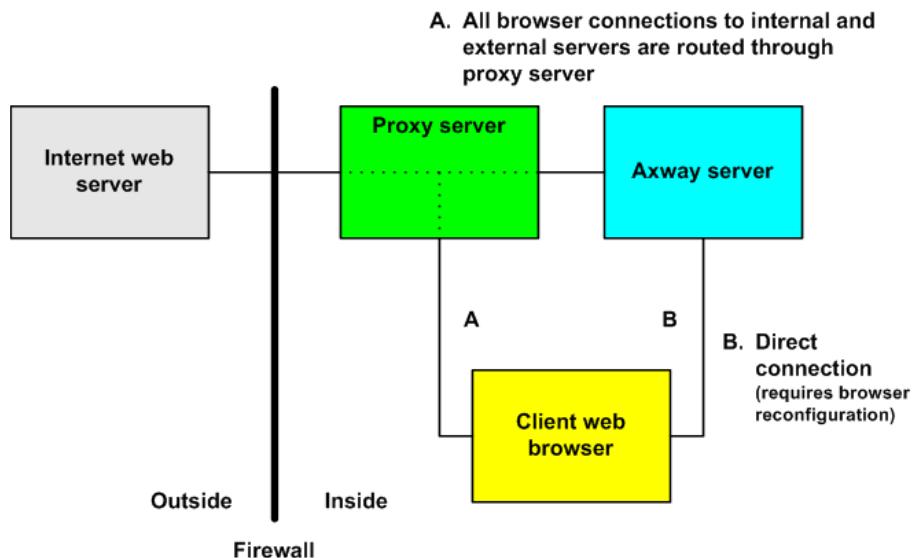
## Forward proxy

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 10 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 10), but this probably would go against policy.

The web proxy server might be set up inside the firewall, as in Figure 10, 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.

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 Activator domains. It also must be configured to provide direct access to internal web servers.

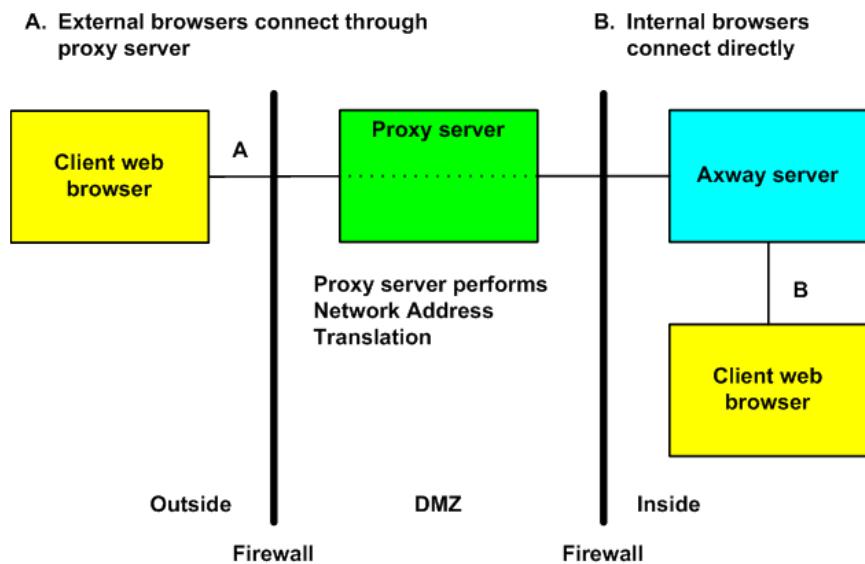


**Figure 10. Forward proxy configuration**

## Reverse proxy

In a reverse proxy configuration, the web proxy server is in the DMZ (Figure 11). 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 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`.



**Figure 11. 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 [Synchrony CSOS](#) on page 607.

## Change user: *root administrator*

The screenshot shows a user interface for managing a user account. At the top, there is a navigation bar with tabs: General (which is selected), Alternate contact, Roles, Personal certificates, and Time zone. Below the tabs, the user's current information is displayed:

User ID:	admin
Current login:	Feb 16, 2007 11:34:48 AM MST
Last successful login:	Feb 15, 2007 11:15:51 AM MST

Below this, there are input fields for the user to change their contact information:

User name:*	root administrator
Email address:*	[Empty]
Phone number:	[Empty]

At the bottom of the form, there is a checkbox labeled "Send an e-mail notification when a document submitted by this user is rejected".

At the very bottom of the form, there is a blue "Save changes" button.

**Figure 12. Change user page in users and roles area of the user interface**

### Concepts

- ◆ [The admin user](#) on page 78
- ◆ [Global user settings](#) on page 85

### Procedures

- ◆ [Add, change, delete users](#) on page 78
- ◆ [Change password](#) on page 79
- ◆ [Add, change, delete roles](#) on page 79
- ◆ [Unlocking a blocked user](#) on page 88

## 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 [Synchrony CSOS](#) on page 607.

### 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. Use the Roles tab to add or change a role for a user.
- ◆ A user has a choice of the time zone to display in the user interface. The default is the server's time zone. See the Time zone tab for other options.
- ◆ 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 can change his own password. In addition, a user with permissions for managing users can change the passwords of others as well as his own password.

To change a password, select **Users and roles > Change my user account**. Or, if you are an administrator, select **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.

All users must have an assigned role. Users can be assigned to multiple roles at the same time, but managing this can be complicated unless role permissions are defined clearly.

### 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**.

## **Role permissions**

When adding or changing a role, you can choose what type of tasks users assigned to the role are allowed to perform. This is done by specifying permissions on the Permissions tab. The more permissions selected, the broader the authority. For an administrator role, it is typical to grant all permissions.

The following list describes all possible permissions for roles. Depending on your user license, not all of these may be available in the user interface or applicable to your users.

### **Approve CSOS orders**

Allows outbound CSOS orders to be signed and verification of inbound signed CSOS orders. This permission only applies for licenses with CSOS authorization for the trading engine.

### **Execute reports**

Allows generating reports.

This permission has the following child permission, which applies only if the parent permission is selected.

**Manage reports.** Allows using the alert activity report.

## Export private keys

Allows users of the trading engine to export private encryption keys when exporting certificates to files. See [Export a certificate to a file](#) on page 424.

## Manage tasks

Allows display of the task list on the home page.

## Search for messages processed by the trading engine

Allows using Message Tracker to search for and view traded messages and information about messages.

This permission has the following child permissions, which can be applied when the parent permission is selected.

**Add notes to messages.** Allows users to write notes while reviewing message details in Message Tracker.

**Delete messages** Allows deleting messages in Message Tracker.

**Manage global message search settings.** Allows changing settings and values on the Message Tracker global settings page.

**Manage document types.** Allows adding document types as a way to identify documents.

**Restrict to selected partners.** Allows limiting the role to specific partners. Users with this role only can search for and view data related to the specified partners. See [Partner restrictions for roles](#) on page 83. For unrestricted access, do not select this.

**Resubmit messages.** Allows resubmitting messages in Message Tracker.

**Save, share, change and delete searches.** Allows managing saved searches.

**View payloads and backups.** Allows viewing backed-up copies of traded messages in Message Tracker.

## View system status

Allows viewing the system management area of the user interface.

This permission has the following child permission, which can be applied when the parent permission is selected.

**Manage system properties and security.** Allows using controls in the system management area of the user interface.

## View trading configuration

Allows viewing community profiles in the trading engine.

This permission has the following child permissions, which apply only if the parent permission is selected.

**Manage partners configuration.** Enables adding, changing and deleting partner profiles. This permission also is necessary to do peer network tasks. Selecting this permission in combination with **Restrict to selected partners** takes away the ability to add and delete partner profiles.

**Restrict to selected partners.** When **Manage partners configuration** also is selected, allows limiting the role to specific partners. Users with this role only can view all partner profiles, but can change data related only to the specified partners. See [Partner restrictions for roles](#) on page 83. Also, users with this role cannot add or delete any partners. For unrestricted access, do not select this.

**Manage trading configuration.** Enables adding, changing and deleting community profiles. This permission also is necessary to do peer network tasks.

## View user activity

Allows viewing of user and roles area of the user interface.

This permission has the following child permission.

**Manage users and roles.** This is the single most powerful permission. A user assigned to a role with this permission has unlimited permissions, regardless of other permissions selected. When this permission alone is selected, users with this role can perform all tasks and navigate to all areas in the user interface. Select this permission only for a role for administrators. (This permission does not enable use of Pedigree Viewer. You must separately apply those permissions.)

## Partner restrictions for roles

When adding or changing a role, you can give users broad or narrow access to partners of your Activator communities. This is done by specifying partners on the partner restrictions tab. You can define by role the partner data users can search for and view in Message Tracker. For users with administrator authority, you can limit the partner profiles the administrators can change.

### Change role: *TestRole*

Role name: \*

Description:

**Permissions | Partner restrictions**

Pick the communities, categories or partners to which you would like to limit this role

Limit this role to the partner selections below  
 Allow this role access to all partners except for those selections below

**Communities | Categories | Partners**

Limit this role based on the members of the selected communities.

Worldwide Trading

**Save changes**

**Figure 13. Partner restrictions tab**

Partner restrictions take effect when **Restrict to selected partners** is selected on the Permissions tab under one or both of the following role permissions:

- ◆ Search for messages processed by the trading engine
- ◆ Manage partners configuration

The conditions set on the Partner restrictions tab apply equally to both of these role permissions. If you want partner restrictions to be identical for searching in Message Tracker and managing partner profiles, select both. If you want partner restrictions to be different for both permissions, set up two roles.

The Partner restrictions tab has two general conditions. These can be further refined with other filtering conditions on the tab's three sub-tabs.

The general conditions are:

**Limit this role to the partner selections below**

Users assigned to the role can access information only for the partners selected on the sub-tabs. Select this if you want the role to have access only to a limited number of partners. Specify the exceptions on the Communities, Categories or Partners sub-tabs.

If you select this and do not specify any exceptions on the sub-tabs, the effect is the role does not have access to any partners.

**Allow this role access to all partners except for those selections below**

Users assigned to the role can access information for all partners with the exception of the partners selected on the sub-tabs. Select this if you want the role to have access to many partners, except for a specified few. Specify the exceptions on the Communities, Categories or Partners sub-tabs.

If you select this and do not specify any exceptions on the sub-tabs, the effect is the role has access to all partners.

You can use one or more of the Partner restriction tab's sub-tabs to select partners. The effects of the sub-tabs are cumulative. For instance, if you select community A on the Communities sub-tab and partners C and D on the Partners sub-tab, all partners of community A as well as partners C and D — regardless whether C and D belong to community A — are affected.

If you do not make any selections on the sub-tabs, this has the effect of denying access to all partners.

If you need help setting up partner categories to use the Categories sub-tab, see [Partner categories](#) on page 182.

## **Managing multiple partner-restricted roles**

The system allows assigning multiple roles to a user. In the case of roles with partner restrictions, limits are applied in an cumulative sense. For instance, as the system builds the list of allowed partners, the roles only can add partners to the list, but not take them away. This way the roles can be applied in any order. The examples in the following table show the effect of assigning multiple roles with different partner restrictions.

Partner	Role 1 restrictions	Role 2 restrictions	Result
Partner A	allows	does not allow	access granted
Partner B	does not allow	allows	access granted

Partner	Role 1 restrictions	Role 2 restrictions	Result
Partner C	does not allow	does not allow	access denied
Partner D	allows	allows	access granted

# Global user settings

**Change global 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 88.

### 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.

#### **Allow browsers to remember user IDs**

Select this to display a check box for **Remember my user ID** on the log-on page. This gives users the option of having their browsers remember their user IDs the next time they log on.

Clear the check box on the global settings page if you do not want browsers to remember user IDs. After disallowing browsers from remembering user IDs, users may not notice the change until logging on for the second time following the change.

## **User security tab**

If 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. The largest allowed value is 20.

If a value of 0 is used, the minimum change count for password re-use is disabled. This means a minimum change count does not affect password re-use.

#### **Elapsed days before password can be reused**

The number of days that must pass before a user can re-use a password. The largest allowed value is 9999.

If a value of 0 is used, elapsed days before a password can be re-used is disabled. This means a password can be re-used immediately if the minimum change count also is 0.

#### **Days password remains valid before it must be reset**

The number of days a password is valid before it must be changed. The largest allowed value is 9999.

If a value of 0 is used, a password remains valid forever.

#### **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. The largest allowed value is 9999.

If a value of 0 is used, a user remains active forever, regardless how much time has elapsed since the user logged on.

#### **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.

#### **Passwords must have at least one upper-case letter and one lower-case letter**

Forces users to have at least one upper-case letter and one lower-case letter in passwords. With or without this selected, passwords are case sensitive.

**Passwords must have at least one number (0 to 9)**

Forces users to include at least one number in passwords.

**Passwords must have at least one special character from the set**

Forces users to have at least one special character in their passwords. Type the permitted characters in the special characters allowed field. For example, you can allow characters such as:  
`~!@#\$%^&\*()=-[]{}\\|;,.;<>?.

## 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.

**Note:** In rare cases the utility may not work. If this happens, try enclosing the password in quotes. For example: “password”.



# 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 121.

## Concepts

- [System monitoring in the user interface](#)
- [External monitoring of server status](#) on page 93
- [Log file tracking](#) on page 94
- [Troubleshooting with log4j file](#) on page 98
- [Send log files to technical support](#) on page 103

## Procedure

- [Real-time viewing of log files](#) on page 96
- [Set up tail as a Windows option](#) on page 97

## System monitoring in the user interface

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 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.

See the following topics for information about other links on the page.

- [Certificate revocation lists](#) on page 437
- [Configuring external SMTP server](#) on page 23
- [Changing search default settings](#) on page 508
- [Binding network interfaces](#) on page 53
- [Statistics monitor](#)

## ***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.

The statistics monitor is intended for advanced users or developers with Java experience. You may find parts of the statistics monitor useful, for example, while working with technical support to troubleshoot an issue.

To explore the statistics monitor and its options, 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. Place your cursor over an option to display a pop-up message that describes the item.
- ♦ Some items in the Available list 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.

The following briefly describes the tabs and charts that display by default on the statistics monitor. Depending on your user license, some of these may not display or apply to you.

### **System health tab**

This tab has two charts:

**Free memory** charts the heap size for the application. This is the memory usage for the application, not for the machine on which the application is running. A see-saw chart pattern is normal and should be expected.

**Database response** reports the interval for the database to response to pings sent at regular intervals. Generally speaking, the faster the response, the better the performance.

### **Internals tab**

This tab has two charts:

**Thread pool** shows active threads per processing node.

**Connection pool** shows active connections to the database.

### **Alert system**

This tab has two charts:

**Current activity** shows counts of alert system activity and per-minute averages of certain high volume counters related to alert processing.

**Cumulative history** shows several long-running counters of total activity in the cluster since the server was started. These only are for current cluster up-time and not since the application was installed.

#### **Message consumption tab**

The chart on this tab reports message consumption per processing node per hour for integration pickup and trading delivery exchanges. When the trading engine is running with multiple nodes or in a multiple-machine cluster, this shows whether the processing load is balanced across nodes.

#### **Message processing tab**

The chart on this tab shows the types of messages the trading engine consumers per processing node per hour. Typically, the message types are XML, EDI or binary.

#### **Message production tab**

This tab has two charts:

**Producer tasks** monitors messages the trading engine has processed but not yet produced.

**Production backlog** monitors messages the trading engine has consumed but not yet processed or produced.

## **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 495.

## 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. Network routing devices such as load balancers, 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 Gateway 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 156.

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 183).

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 105.

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. 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 98 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 Activator software is installed. The names have the following format:  
**hostname\_cn\_access.log.oooooooo1**. The **cn** stands for control node. The trailing number (oooooooo1) identifies rolling logs. Once a log reaches a certain size, events write to another log (oooooooo2). 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

Other log files include Axway\_InstallLog.log, which contains information about the initial system installation, and db\_configurator.log, which contains messages logged by the database configuration utility.

Axway\_InstallLog.log writes to [install directory]\[build number]\logs\install. The db\_configurator.log writes to [install directory]\[build number]\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 logfile**

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 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 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. 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. You do need to restart the server if you add or delete a property.

---

**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.

**Note:** If your user license supports ePedigree functionality, you can add the following property to the log4j.properties file to generate verbose events:

```
log4j.category.com.cyclonecommerce.epedigree=debug
```

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.collaboration,peernetwork=info**

Messages related to the peer network. These events are generated only if your user license enables peer network functionality.

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

Messages related to Alert system activity

## Trading engine messages

### **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.

**log4j.category.com.cyclonecommerce.tradingengine=info**

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

## 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.http.server=info**

Controls logging for code to interface with the embedded HTTP server.

**log4j.category.com.axway.transport.http.server=info**

Controls internal logging for the embedded HTTP server.

**log4j.category.org.mortbay=info**

**log4j.category.com.cyclonecommerce.tradingengine.transport.ft  
p.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.ft  
p=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.ft  
p.server=info**

Controls logging for code to interface with the embedded FTP server.

**log4j.category.org.apache.ftpserver=info**

Controls internal logging for the embedded FTP server.

**log4j.category.com.cyclonecommerce.tradingengine.transport.e  
mail.smtp=info**

Messages related to SMTP delivery exchange.

**log4j.category.com.cyclonecommerce.tradingengine.transport.e  
mail.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
log4j.category.net.sf.ehcache.store.DiskStore=fatal
```

## Send log files to technical support

Activator has an option in the user interface for bundling the application's log files in one compressed file and sending the packaged files to technical support. When working with users to resolve issues, technical support often asks for log files to help in troubleshooting. This option makes it easy to supply the latest log files.

To use this option, the machine running Activator server must have an active Internet connection, as the log files are sent via HTTP.

To send logs to technical support through the user interface, select **Help > Send logs**.

### Send logs to support

Use this dialog to send your current log and configuration files to Support for analysis. For identification purposes, the company name from your license file will automatically be sent as well.

You may optionally include a description containing information such as your phone number, email address, and support ticket number (if you have one). Any other useful information such as the conditions leading up to the problem, and when it began, would be helpful.

Description:

If a proxy is required to make outbound connections, specify the community whose proxy definition should be used.

Proxy:

Perform thread dump (places output in each node log)

**Figure 14. Send logs to support page**

The following are the fields on the send logs page.

### Description

Type pertinent information to identify yourself, your organization and the issue you want resolved. Other information is helpful, such as the version of Activator you are using, along with the operating system and database type. You also can describe the sequence of events or actions leading to the problem.

If you have not opened a support ticket for the issue, go to <http://support.cyclonecommerce.com> and open one. Or, send a message to support at [support2@us.axway.com](mailto:support2@us.axway.com). If you already have an open ticket, include the number in the description.

### Proxy

This field is active only if your community routes outbound HTTP messages through a proxy. If the field is active, select a community. If you are not sure whether a proxy is required, ask your network administrator. See [Outbound HTTP proxy](#) on page 377 for more information.

### Perform thread dump (places output in each node log)

Provides a snapshot of all threads in the Java virtual machines on the running nodes. This check box is selected by default. You can leave the thread dump turned on, unless technical support advises otherwise.

If the option for sending log files to technical support is not available because it is impractical or not possible to access the user interface, there is a command-line tool you can use instead. See [diagnose](#) on page 46.



# 9 Events system

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 126.

[The events.xml file](#) on page 106 describes how events are published to log files and JMS routers.

[The alerts.xml file](#) on page 121 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](#)
- ◆ [The events.xml file](#) on page 106
- ◆ [Log file event router](#) on page 112
- ◆ [JMS event router](#) on page 114
- ◆ [Oracle AQ event router](#) on page 114
- ◆ [XML for JMS and AQ event routers](#) on page 114
- ◆ [The alerts.xml file](#) on page 121
- ◆ [Event tables](#) on page 126

## 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 94). 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 Activator, are named in the following format: hostname\_te\_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.

By 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 15 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 112 and [JMS event router](#) on page 114.

```
- <EventRouters>
+ <!-->
- <EventRouter id="Important Events to Log File" class="com.cyclonecommerce.events2.router.LogFileRouter" active="true">
  <Parameters file=".\\logs\\%nodeName%_events.log" rollOnStart="false" autoFlush="true" maxFileSize="2M" maxBackupFiles="5"/>
  <MetadataProcessorListRef ref="Messaging"/>
  <EventFilterRef ref="Important"/>
</EventRouter>
- <EventRouter id="Message Detail to Log File" class="com.cyclonecommerce.events2.router.LogFileRouter" active="false">
  <Parameters file=".\\logs\\%nodeName%_message_detail.log" rollOnStart="true" autoFlush="true" maxFileSize="2M" maxBackupFiles="5"/>
  <MetadataProcessorListRef ref="Messaging"/>
  <EventFilterRef ref="Message Detail"/>
</EventRouter>
- <EventRouter id="Message Summaries to Log File" class="com.cyclonecommerce.collaboration.events.MessageSummaryLogFileRouter" active="false">
  <Parameters file=".\\logs\\%nodeName%_message_summary.log" rollOnStart="true" autoFlush="true" maxFileSize="2M" maxBackupFiles="5"/>
  <MetadataProcessorListRef ref="Messaging"/>
  <EventFilterRef ref="Message Summary"/>
</EventRouter>
+ <!-->
+ <!-->
+ <!-->
+ <!-->
+ <!-->
+ <!-->
+ <!-->
+ <!-->
+ <!-->
+ <!-->
+ <!-->
+ <!-->
+ <!-->
+ <!-->
+ <!-->
+ <!-->
+ <!-->
</EventRouters>
```

Figure 15. 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 **MetadataProcessors** section follows the **EventRouter** section of the events.xml file. Figure 16 is an example of the default MetadataProcessors section.

```
- <MetadataProcessors>
-   <!--
-     Do NOT remove or change the following MetadataProcessorList definition!
-   -->
-   <MetadataProcessorList id="Messaging">
-     <MetadataProcessor type="CSOS_Messaging" class="com.cyclonecommerce.collaboration.events.MessagingMessageMetadataProcessor"/>
-     <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 16. 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 17 is a partial example of the default EventFilters section.

```
- <EventFilters>
+ <!-->
- <EventFilter id="Integrator">
- <OrFilter>
<EventTypeFilter type="Messaging.Transport.ReceiveFailure"/>
<EventTypeFilter type="Messaging.Packaging.DecryptionFailure"/>
<EventTypeFilter type="Messaging.Packaging.SignatureVerificationFailure"/>
<EventTypeFilter type="Messaging.Packaging.CertificateValidationFailure"/>
<EventTypeFilter type="Messaging.Packaging.DecompressionFailure"/>
<EventTypeFilter type="Messaging.Message.Collaboration"/>
<EventTypeFilter type="Messaging.Message.MessagePackaged"/>
<EventTypeFilter type="Messaging.Transport.SendFailure"/>
<EventTypeFilter type="Messaging.Transport.OutputTransferCompleted"/>
<EventTypeFilter type="Messaging.Transport.RetryFailure"/>
</OrFilter>
</EventFilter>
- <EventFilter id="Messaging To Database">
- <AndFilter>
<EventTypeFilter type="Messaging"/>
- <OrFilter>
<EventLevelFilter level="High"/>
<EventLevelFilter level="Warning"/>
<EventLevelFilter level="Error"/>
</OrFilter>
</AndFilter>
</EventFilter>
```

**Figure 17. 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 subtypes. The type attribute must be specified and its value must be the name of an event type defined in [Event tables](#) on page 126

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 94](#)). 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 106](#).

```
<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 definition.

### 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 Axway 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 Axway 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.

### **BackupFilename**

Name of the message backup file.

### **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.

### **CompressionType**

The type of compression used to compress a message.

### **ConsumptionFilename**

Name of the file consumed from the pickup exchange point.

### **ContentMic**

MIC value of the message.

**ContentMimeType**

The MIME type of the message payload.

**CoreId**

Unique message identifier within the trading engine.

**DigestAlgorithm**

Digest algorithm used for the 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.

**DocumentId**

Control ID if the message payload is EDI.

**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.

**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.

**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.

**ebXMLMessageId**

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

**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.

### **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.

### **EncryptionAlgorithm**

Encryption algorithm used for the message.

### **IntegrationId**

Used to attach customer-specific ID to a message.

### **IsFinalState**

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

### **IsResponsePositive**

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

### **MaxRetries**

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

### **MaxSendAttempts**

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

### **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.

### **MessageId**

Business protocol message ID.

### **MessageState**

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

**MicCheckFailed**

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

**NextRetryNum**

The next retry increment.

**NextRetryTime**

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

**PeerAddress**

URL of a peer in a peer network.

**ProductionFilename**

Name of the file produced to the delivery exchange point.

**ReceiptMicAlgorithm**

MIC algorithm used for the receipt.

**ReceiptRequested**

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

**ReceivedContentMic**

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

**ReceiverRoutingId**

Routing ID of the receiving party.

**RefToCoreId**

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

**RejectionDescription**

Rejection reason for a rejected message.

**ResendInterval**

The elapsed time to wait between resend attempts.

**ResponseCoreId**

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

**SendAttempt**

The number of sends attempted.

**SenderRoutingId**

Routing ID of the sending party.

**TransportType**

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

## **Persistence event router**

Up until version 5.4.2, the router to persist events to the database was active in the events.xml file. Effective with version 5.4.2, the event persistence router is inactive by default. A change in the way the system reports events makes it possible to no longer have to store events in the database, freeing database space and reducing data traffic.

The persistence event router remains in events.xml, but is commented out and inactive (Figure 18).

---

```
<!--
<EventRouter id="Message Events to Database"
  class="com.cyclonecommerce.events2.router.PersistenceRouter"
  active="true"
  priority="2147483647">
  <EventFilterRef ref="Messaging To Database"/>
</EventRouter>
```

---

**Figure 18. Persistence event router in events.xml**

We recommend keeping the router inactive. The cases for activating the router are rare (for example, a third-party back-end system retrieves event data from the database). Before activating, consult technical support.

# 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 23.
  - ◆ 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 19 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 19. 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 **{AlertEmailAddresseses}** 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 125

### 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.NumberOfOccurrences	Number of times the event has been published

The following meta-data are available to events at the Messaging level (see [Messaging](#) on page 127) 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.EbxmService  
Message.EbxmAction  
Message.EbxmCpaId  
Message.EbxmConversationId  
Message.EbxmRefToMessageId  
Message.EbxmMessageId  
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 127 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 127, [Transport](#) on page 131 and [Packaging](#) on page 133.

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

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

<b>Event</b>	<b>Lvl</b>	<b>Description</b>
Duplicate	Warn	Received a duplicate message
Duplicate.Message	Warn	Received a duplicate message
Duplicate.Payload	Error	Received a duplicate document
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
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

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

Event	Lvl	Description
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
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

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

Event	Lvl	Description
ResendsExhausted	Error	Message resent maximum times
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
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

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

Event	Lvl	Description
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
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 encrypted
ValidationFailure.NotSigned	Error	An inbound message failed because it was not signed

## 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
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

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

Event	Lvl	Description
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

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

Event	Lvl	Description
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
SignatureVerificationInitiated	Low	A signature verification and validation has started
UnpackageCompleted	Low	Unpacking completed
UnpackageInitiated	Low	Unpacking 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
NotPeer	Error	Either the sender or receiver of a peer message is not a 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 121.

**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.CertificateMetadataModified	High	The meta-data on a certificate has been created, update or deleted.
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.PsePolicyUpdate	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
CertificateManagement.SslCertificateNotYetValid	High	An SSL certificate precedes its validation date

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

<b>Event</b>	<b>Lvl</b>	<b>Description</b>
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
Collaboration.BuildingBlockAdded	High	A message handler condition has been added
Collaboration.BuildingBlockRemoved	High	A message handler condition has been removed
CpaManagement.CpaExpired	High	A community has a CPA that will expire in two weeks or less.

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

Event	Lvl	Description
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
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

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

Event	Lvl	Description
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.

CSOS-related events provide extended CsosMetadata content for one or more of the following:

- ◆ CsosDeaRegistrationNumber
- ◆ CsosPoNumber
- ◆ X509Certificate

X509Certificate is the CSOS signing certificate.

These meta-data are reported when available. For example, if a CSOS order verification failed event occurs because the signing certificate is not trusted, the DEA number and purchase order number are not available because the CSOS signature could not be validated. In this case, the event contains only the certificate meta-data.

In addition to the events in the following CSOS table, these other events also contain extended CsosMetadata content if generated when CSOS messages are handled:

- ◆ Messaging.Message.MessageRejected
- ◆ Messaging.Message.Duplicate.Payload

**Table 13 - CSOS events.**

Event	Lvl	Description
CSOS.Messaging.Approval	Low	A user has signed and approved an outbound message
CSOS.Messaging.Approval.Rejected	High	A message awaiting signing was rejected
CSOS.Messaging.QueuedForApproval	Low	A document is waiting for user signing

**Table 13 - CSOS events.**

Event	Lvl	Description
CSOS.Messaging.Verification	Low	This event has been deprecated.  The user signature of an inbound message has been verified
CSOS.Messaging.Verification 2	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

## ePedigree

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

**Table 14 - ePedigree events**

Event	Lvl	Description
Pedigree.DigSign.Error	Error	An exception occurred in signing a pedigree message, verifying the signature or when a signature check fails for an inbound message.
Pedigree.Formation.Error	Error	Unable to transform a message from integration into a pedigree due to missing data.
Pedigree.Management.ManualSign	High	A user of Pedigree Viewer has signed a pedigree. The event includes the serial number of the pedigree and the ID of the user who performed the action.
Pedigree.Management.Void	High	A user of Pedigree Viewer has canceled a pedigree. The event includes the serial number of the pedigree and the ID of the user who performed the action.
Pedigree.Messaging.Failure	Error	An exception occurred in processing a pedigree message, either an integration message or a message from a partner.

**Table 14 - ePedigree events**

Event	Lvl	Description
Pedigree.Shipping.Error	Error	A failure occurred in attempting to send an outbound pedigree. For example, trying to send to a routing ID that does not exist.
Pedigree.Validation.Error	Error	A pedigree message failed validation to the customer pedigree schema or the EPCglobal schema.
Pedigree.Workflow.Completed	High	A pedigree-related message entered the system and was processed successfully.

## CEM

The following are events related to certificate exchange messaging (CEM).

**Table 15 - CEM events**

Event	Lvl	Description
CEM.CertificateInstall.Failure	Error	Installation of community certificate through CEM failed
CEM.CertificateInstall.Success	Low	Installation of community certificate through CEM succeeded
CEM.Message.Failed	Error	Processing of received EDIINT CEM message failed
CEM.Message.Invalid	Error	Received EDIINT CEM message was invalid
CEM.Request.Accepted	Low	Received EDIINT CEM trust request was accepted
CEM.Request.Accepted_Failed	Error	Failed to accept received EDIINT CEM trust request
CEM.Request.Failed	Error	Processing of EDIINT CEM request failed
CEM.Request.Invalid	Error	Received EDIINT CEM request was invalid
CEM.Request.Received	Low	EDIINT CEM request was received

**Table 15 - CEM events**

<b>Event</b>	<b>Lvl</b>	<b>Description</b>
CEM.Request.Rejected	High	Received EDIINT CEM trust request was rejected
CEM.Request.Rejected_Failed	Error	Failed to reject received EDIINT CEM trust request
CEM.Request.Sent	Low	EDIINT CEM request was sent
CEM.Response.Accepted	Low	Received EDIINT CEM trust response accepted previously sent trust request
CEM.Response.Failed	Error	Processing of received EDIINT CEM trust response failed
CEM.Response.Ignored	Low	Received EDIINT CEM trust response was ignored
CEM.Response.Received	Low	EDIINT CEM response was received
CEM.Response.Rejected	High	Received EDIINT CEM trust response rejected previously sent trust request
CEM.Response.Sent	Low	EDIINT CEM response was sent

## WebTrader

The following are events related to WebTrader.

**Table 16 - WebTrader events**

<b>Event</b>	<b>Lvl</b>	<b>Description</b>
WebTrader.Document.Downloaded	Low	A web trader has clicked a document link to download it to his local hard drive.
WebTrader.Document.Downloaded.Viewed	Low	A web trader has clicked a document link to view a document in the browser.
WebTrader.Document.Deleted	Low	A web trader has deleted a document.

## 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.

In the peer network, when a receipt is received by a peer who did not send the message the receipt acknowledges. The peer ignores the receipt, but sends a new peer message, whose packaged content is the receipt. This ensures the receipt properly finds its way to the peer who originated the outbound message to the partner.

## 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 17 - 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

**Table 17 - Administration system events. Prefix: Administration.System.**

Event	Lvl	Description
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 18 - Administration licensing event**

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

## Administration persistence

**Table 19 - 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 20 - Security events. Prefix: Security.**

Event	Lvl	Description
Authentication.AuthenticationFailure	High	A user cannot be authenticated
Authentication.AuthenticationSuccess	High	A user has been authenticated

**Table 20 - Security events. Prefix: Security.**

<b>Event</b>	<b>Lvl</b>	<b>Description</b>
Authentication.LockedOutUserAttempt	High	A user retries 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
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 with Activator

Activator 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 Activator supports for exchanging messages with partners. User-defined message routing and rules-based processing can be handled at the partner or document level.

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](#)
- [Major components](#) on page 148
- [Trading engine glossary](#) on page 149
- [Configuration outline](#) on page 151
- [Interoperability](#) on page 153
- [Security guidelines](#) on page 153
- [Maintenance tips](#) on page 154

## How the trading engine processes

Figure 20 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.

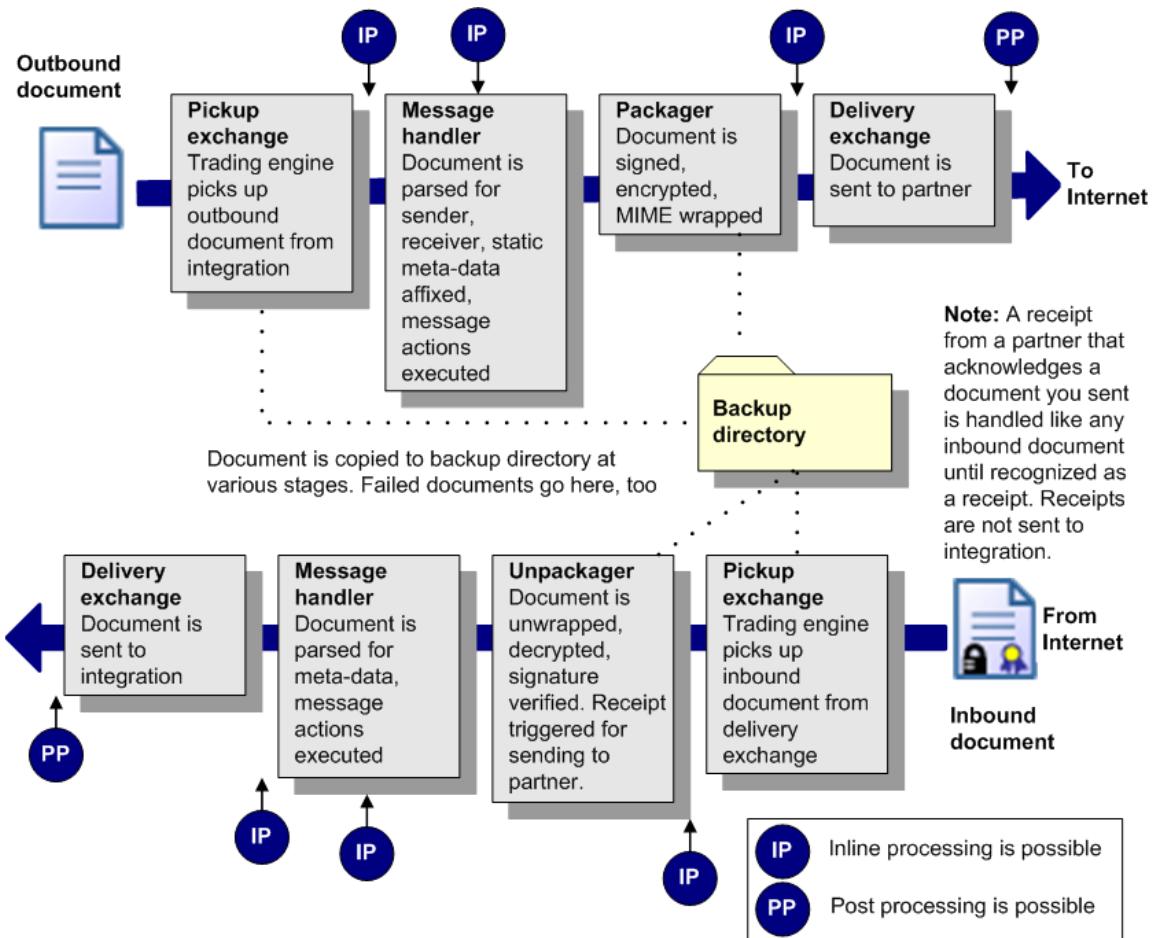


Figure 20. Outbound and inbound processing flow

## 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 <b>key</b> .
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 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 [Installing and starting the server](#) on page 15.

- 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 155 for information about community and partner profiles and the roles they play in Activator. See [Add a community](#) on page 160 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 397.
- 3** Determine whether you want to complete any other configuration for your community profile.  
See [Check list for community configuration](#) on page 160.
- 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 171.
- 5** Export your community profile and distribute it by a secure means to your trading partners. See [Export a community profile](#) on page 167.

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 162 and [Add a partner](#) on page 164.
- 7** Perform test trading to validate your configuration.

See [Test trading](#) on page 529.

- 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 495.

## Interoperability

The trading engine has been certified for interoperability for AS1, AS2, AS3 and ebXML. See [www.ebusinessready.org](http://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 94.
- ◆ 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 515.
- ◆ 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 relationship 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

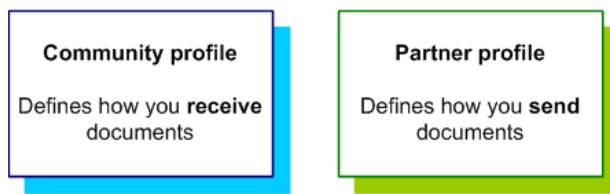
- ◆ [How profiles work](#)
- ◆ [The community profile on page 156](#)
- ◆ [Anatomy of a community profile on page 157](#)
- ◆ [Add a community on page 160](#)
- ◆ [Check list for community configuration on page 160](#)
- ◆ [The partner profile on page 162](#)
- ◆ [Anatomy of a partner profile on page 163](#)
- ◆ [Add a partner on page 164](#)
- ◆ [Searching for partners on page 165](#)
- ◆ [Routing IDs on page 165](#)
- ◆ [Firewalls and proxy servers on page 171](#)
- ◆ [Self-registration of AS1, AS2 partners on page 175](#)
- ◆ [Partner data form on page 178](#)
- ◆ [Partner categories on page 182](#)

## Procedure

- ◆ [Exporting and importing profiles on page 166](#)

## How profiles work

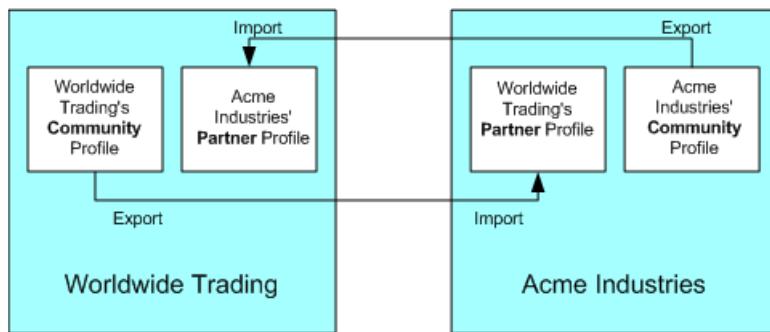
The trading engine organizes the information needed to exchange documents with partners in community and partner profiles. The use of profiles makes it easy to set up and maintain trading relationships. A community profile defines how you receive documents from partners. A partner profile defines how you send documents to a partner. The user interface guides you through most steps for setting up community and partner profiles.

**Figure 21. Purpose of community and partner profiles**

If you and your partner use Axway 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 22 shows how community profiles are exchanged between partners. This example shows two companies exporting community profiles to be exchanged and imported as partner profiles.

**Figure 22. 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 by partners to encrypt messages before sending.

In versions of 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 23) 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 around that part of the graphic in the user interface. Click inside the red box to go to the page the label references.



**Figure 23. 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 of 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 397.

### 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 165.

### 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.

## FTP users

The FTP users page lists user accounts associated with embedded FTP servers and the usage for the users. For more information see [Managing users of embedded FTP](#) on page 329.

The FTP users icon appears only when an embedded FTP server has been configured for a community.

## 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 206.

## 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 206.

## 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 487.

## Message validation

The message validation page lets you set whether a community accepts or rejects EDI messages with duplicate control IDs. You also can specify whether a community accepts or rejects signed or encrypted messages. For more information see [Inbound message validation](#) on page 481.

## 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 447.

### **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 203.

### **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 377.

### **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 information. For a list of all components that make up a community profile, see [Anatomy of a community profile](#) on page 157

### **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 169.

## **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 165.

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 559.

- 2** What kind of certificate do you want to use: self-signed or CA? See [Manage certificates](#) on page 397.
- 3** What integration exchange do you want to use for picking up messages from a back-end system? See [Integration exchanges](#) on page 206.
- 4** What trading exchange do you want to use for receiving messages from partners? See [Trading delivery exchanges](#) on page 203.
- 5** What integration exchange do you want to use for routing messages received from partners to a back-end system? See [Integration exchanges](#) on page 206.
- 6** Have you added a trading partner, either by importing a profile file or manually configuring a profile? See [Add a partner](#) on page 164.
- 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 171 and, if applicable, [Outbound HTTP proxy](#) on page 377
- 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 447.
- 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 448.
- 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 481.
- 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 265.

- 12** Do you want to set up conditions for copying or rejecting messages?  
See [Message handling](#) on page 487.
- 13** Have you configured the global external SMTP server for sending e-mail messages? See [Configuring external SMTP server](#) on page 23.
- 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 515.

## The partner profile

Just as you give a partner your community profile to use as a partner profile, 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 Axway 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 24) 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 around that part of the graphic in the user interface. Click inside the red box to go to the page the label references.



**Figure 24. 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 182.

### 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 203.

### 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 397.

## 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 165.

## 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.

## FTP users

The FTP users page lists user accounts associated with embedded FTP servers and the usage for the users. For more information see [Managing users of embedded FTP](#) on page 329.

The FTP users icon appears only when a partner FTP account has been associated with a community delivery exchange.

# Add a partner

There are several ways to add a partner. The profile wizard prompts you for the required information.

## Manually create a new 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 163

### 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 168.

## **Searching for partners**

A search feature is available to help you locate partner profiles. Searching is possible by:

- ◆ Full or partial partner profile names or routing IDs or both. You must use wildcard characters for partial name and routing ID searches.
- ◆ Partners' community membership or partners who do not belong to a community.
- ◆ Partners' delivery message protocol.

You also can specify how much information to display for the partners found in a search. In addition to partner name, you can display default routing ID, contact, default delivery exchange, categories and communities.

To display a panel for typing search conditions for partner profiles, select **Partners > Manage Partners** to open the pick a partner page. This page lists all partner profiles, regardless of community affiliation. The partner search panel on the left side lets you search by name, routing ID or both. To further narrow the search, you also can search by community membership and delivery protocol.

There are two supported wildcard characters for searching by name and routing ID. The characters are \* and ?. The \* is used in place of multiple characters. The ? is used in place of a single character. For example, if searching for the word **wildcard**, you could type **wildc\***, where \* is in place of **ard**. You also could type **wil?ca?d**, where ? is in place of **d** and **r**, respectively.

## **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 exchanged 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 559 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 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 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 178 provides a way for documenting the data needed to create a partner profile.

If you are upgrading from 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](#)
- ◆ [Export a partner profile](#)
- ◆ [Import a partner profile](#) on page 168
- ◆ [Import company profile as community profile](#) on page 169

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 169.

The trading engine exports a community profile as a partner profile XML file. A community profile cannot be exported as a community profile.

## ***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 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.

## ***Export a partner profile***

To export a partner profile to an XML file, click **Partners** on the toolbar and then select a partner. At the bottom of the partner summary page, click **Export this partner's profile**.

A reason to export a partner profile is to create a back-up copy for later importing, if needed.

Partner profiles can be exported with a tool as well as through the user interface. The tool allows exporting partner profiles singly or in a batch. See [exportProfile](#) on page 47.

An exported partner profile contains information such as the partner's name and routing ID, plus the configured transports for sending messages to the partner. If there is more than one exchange, the order of the transports is preserved, as the exchange at the top of the list is the default.

Other preferences included in exported partner profile files include:

- ◆ All advanced settings that display on transport maintenance pages in the user interface, such as maximum concurrent connections, retries, connection, response and read time-outs.
- ◆ HTTP chunking and attempted restarts for HTTP.
- ◆ Transport friendly names.
- ◆ The transport's setting for backing up files.
- ◆ The paths and file names of post-processing scripts, but not the scripts themselves.
- ◆ Information for inline processing Java classes is included in exported profiles, but not the Java classes themselves.
- ◆ If the profile has an FTP transport with an alternate command set file, that preference is included in the exported file, but not the command set file itself.

## ***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 41.

If a partner uses a trading engine other than 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 178 provides a way for documenting the data needed to create a profile.

## **Import company profile as community profile**

You can export company profiles from 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 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 41.

## **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 Interchange or Activator 4.2.x or later or are compatible (see [Creating profiles outside the application](#) on page 170).

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 39.

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 25. 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.

<b>\profiles</b>	<b>\autoImport</b>	<b>\rejected</b>
	<b>\backup</b>	<b>\community</b> <b>\partner</b>
	<b>\staged</b>	<b>\community</b> <b>\partner</b>

**Figure 25. Profile autoImport and related directories**

When exporting a company profile from 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/InterchangePartnerConfig.xsd)

[http://www.cyclonecommerce.com/Schemas/2001/09/  
InterchangeCompanyConfig.xsd](http://www.cyclonecommerce.com/Schemas/2001/09/InterchangeCompanyConfig.xsd)

[http://www.cyclonecommerce.com/Schemas/2001/08/  
CycloneOrganizationProfile.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.

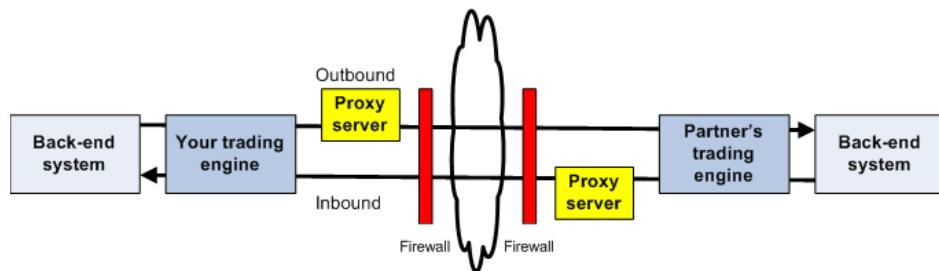
Moreover, your network may require outbound HTTP messages to the Internet to go through a proxy server on your network. On rare occasions, the messages you send may have to go through a proxy server on your partner's network.

---

**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 26 shows where a firewall or proxy server could be located in proximity to your or your partner's network.



**Figure 26. Possible locations for firewalls or proxy servers**

The following are guidelines for outbound and inbound traffic.

## Outbound

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 want 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.

## Inbound

The firewall considerations for inbound traffic are similar to those for outbound traffic. You can allow blanket inbound traffic on a particular port such as 4080 or you can specify per-partner firewall rules based on the IP address of each partner who connects to you. Specifying partner-specific firewall inbound firewall rules provides a high level of protection against denial-of-service attacks. As with partner-specific outbound firewall rules, however, it imposes a firewall maintenance burden if the partner's IP address changes or if you add partners.

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 Activator. For example, if an outbound connection is made to an HTTP host on port 4080, 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.

## ***Activator in a firewall environment***

Figure 27 depicts a standard architecture for deploying 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 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 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.

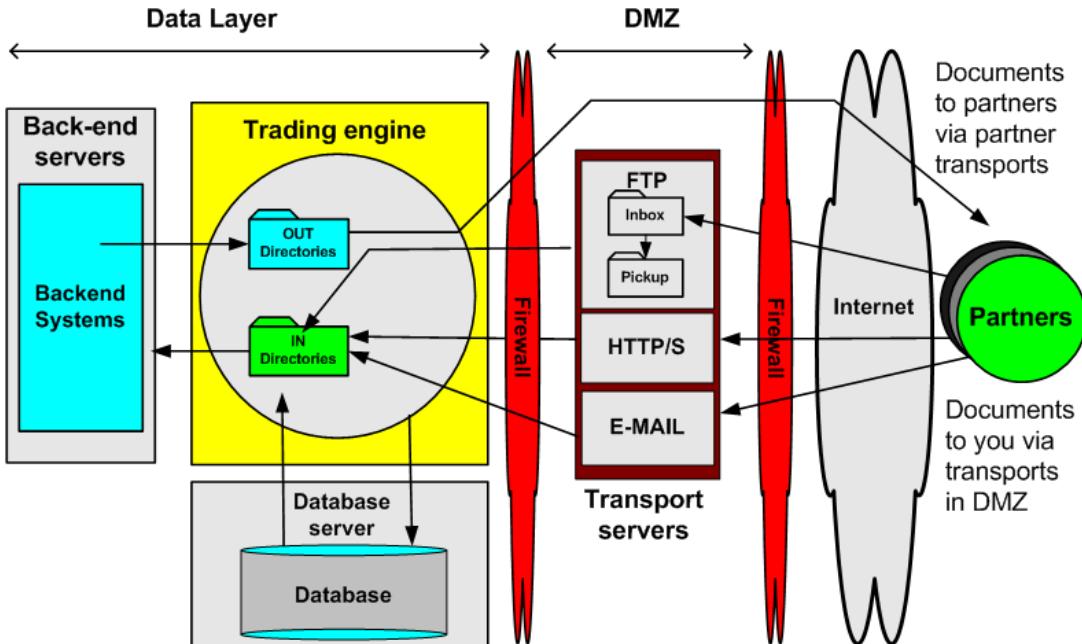


Figure 27. Standard firewall architecture

## ***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 or load balancer, you must make sure your partners have the correct public URL to send you documents. Values such as the host, port and path in the public URL may be different than the internal values because of remapping performed by the firewall or load balancer.

Depending on your transport, your partner needs a URL in the following format:

```
http://[fully qualified domain name or IP address]:4080/  
exchange/[routing ID]
```

```
https://[fully qualified domain name or IP address]:4080/  
exchange/[routing ID]
```

You may have to contact your company's firewall administrator to obtain the correct public URL.

You can change the URL for partners on the transport maintenance page of the user interface. For more information see [HTTP maintenance](#) on page 305. 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.

A similar consideration applies to embedded FTP servers. You must specify the external public host and port in the server settings.

## ***Getting a partner's external connection details***

If you will send documents via HTTP or FTP, contact your partner to determine the correct external host, port and path (if applicable) for connecting to the partner's server. If your partner uses 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 external name or IP address and port number for each transport protocol you intend to use.

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 305.

## 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 377.

# Self-registration of AS1, AS2 partners

The registration wizard helps partners of a trading community generate partner profiles for use by a Gateway 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 577.

A likely use of the registration wizard is for partners who have a trading engine other than Interchange or Activator. The wizard prompts a user to supply the information Gateway Interchange needs to build a valid partner profile. A community, however, could have partners who use 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.



**Figure 28. First page of the partner registration wizard**

The following topics describe how a 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 Interchange or Activator 5.0 or later or 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 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	
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	

Setting	Partner's data
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	
Receiver XPath if custom XML type	

## **Partner categories**

The user interface lets you sort partners into categories. Assigning partners to categories can help you to group partners in a way that suits your needs. For instance, you could add categories representing geographic regions and then assign partners to the applicable regions.

To create a partner category, select **Partners > Manage categories** to open the Pick a category page. Select **Add a category** to open the Add a category page. Type a category name. Optionally, select a parent category. Click **Add** to create the category.

To add a partner to a category, click **Categories** from the partner's summary page to open the Choose categories page. Select the check box for a category and click **Save changes**.



# 12 Embedded transport servers

The trading engine lets communities use embedded servers with some 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 certain delivery exchanges with the delivery exchange wizard (see [Delivery exchange wizard](#) on page 212). 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 201.

## Concepts

- ◆ [Global embedded HTTP server](#) on page 184
- ◆ [Global embedded SMTP server](#) on page 185
- ◆ [Global embedded web services API server](#) on page 186
- ◆ [Embedded HTTP use cases](#) on page 194

## Procedure

- ◆ [Change community embedded server](#) on page 187

## Pages and fields

- ◆ [Community embedded HTTP fields](#) on page 188
- ◆ [Community embedded FTP fields](#) on page 190
- ◆ [Community embedded SMTP fields](#) on page 192

# 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, click **Trading configuration** on the toolbar. On the pick a community page, click the link near the bottom of the page named **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 188).

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)**

The maximum number of seconds the server will wait when reading data from a partner.

**Restartable minimum size (KB)**

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 kilobytes. 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.

# 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, click **Trading configuration** on the toolbar. On the pick a community page, click the link near the bottom of the page named **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 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)

The maximum number of seconds the server will wait when reading data from a partner.

# 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, click **Trading configuration** on the toolbar. On the pick a community page, click the link near the bottom of the page named **Configure the global embedded Web services API server**.

For information about how to configure the server for use in trading, see [Web services API integration](#) on page 255.

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 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 by 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 name of the embedded transport server you want to change.
- 3** Edit the values on the Settings or Advanced tabs as necessary and click **Save changes**. For field descriptions see:
  - ◆ [Community embedded HTTP fields](#) on page 188
  - ◆ [Community embedded FTP fields](#) on page 190
  - ◆ [Community embedded SMTP fields](#) on page 192

## Community embedded HTTP fields

The following are the maintenance fields for a community embedded HTTP or HTTPS transport server.

### Change this embedded transport server

Community: *Worldwide*

Server type: *HTTPS*

Settings Advanced

Name: \*

Host:

Port: \*

⚠ This server does not have an SSL certificate. [Add an SSL server certificate](#)

This server requires client authentication. The partner must present an authentication certificate trusted by the server when connecting.

**Save changes** **Cancel**

Figure 29. Community embedded HTTPS server settings tab

## Settings tab

### 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.

**Add an SSL server certificate**

or

**SSL server certificate**

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 authentication**

Select this to use the partner's certificate to authenticate the partner when the partner connects to the server.

**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)**

The maximum number of seconds the server will wait when reading data from a partner.

### Restartable minimum size (KB)

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 kilobytes. 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.

## Community embedded FTP fields

The following are the maintenance fields for a community embedded FTP transport server.

### Change this embedded transport server

Server type: *FTP*

The screenshot shows a configuration dialog for an FTP server. At the top, there are three tabs: 'Settings' (selected), 'Advanced', and 'Trusted root certificates'. Below the tabs, the 'Name:' field contains 'Trading FTP' and the 'Port:' field contains '4021'. A note below the port says: 'This server requires client authentication. The partner must present an authentication certificate trusted by the server when connecting. Enabling this also disallows clear connections.' Another note states: 'This server does not have an SSL certificate. [Add an SSL server certificate](#)'. A large text area provides instructions about external host and port resolution. At the bottom, there are fields for 'External host or IP address:' (containing 'host.domain.com') and 'External port:' (containing '4021'). At the very bottom are 'Save changes' and 'Cancel' buttons.

**Figure 30. Community embedded FTP server settings tab**

## Settings tab

### 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.

### Port

The port on which the trading engine listens for connection requests.

### This server requires client authentication

Select this to use the partner's certificate to authenticate the partner when the partner connects to the server.

### Add an SSL server certificate

or

### SSL server certificate

For optional SSL, the 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.

If you use a self-signed certificate, it displays on the trusted root certificates tab. A self-signed certificate is a root certificate. For a server certificate issued by a certificate authority, you may also have to use the trusted root certificates tab to import a CA-issued root certificate for the server certificate

### External host or IP address

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. In many cases you must change this to the external name used by your network firewall or load balancer. Contact your network administrator if you need help with this field.

### External port

The port number that a community's partners must use to connect to this embedded server. Contact your network administrator if you need help with this field.

## Advanced tab

### Maximum concurrent connections

The maximum number of simultaneous connections allowed from partners. If the trading engine has multiple processing nodes, each instance of the server running on different computers can host this many connections.

### Read timeout (seconds)

The maximum number of seconds the server will wait when reading data from a partner.

### Allowed passive ports

Ports for inbound passive data connections from FTP clients. You can enter multiple ports. Each port number must be separated by a comma or you can specify a range of ports. The following example uses comma-separated port numbers and a range of ports:

**50000,50001,50010-50020**

If clients are to make secure data connections to the server, these ports must be opened in the firewall. If secure connections are not required, most firewalls are FTP-aware and automatically open the data connection port based on the fact that a client already has a connection to the FTP control port.

If you use a Windows operating system, we strongly recommend disabling the Windows firewall. Use instead an FTP-aware external firewall. Otherwise, you must add an exception for each passive port in the Windows firewall.

In the case of SSL connections, even FTP-aware firewalls cannot dynamically open passive ports. If clients will connect over SSL, open the same range of ports in the firewall as you specify in this field. Failure to do so will result in clients experiencing hangs when performing LIST, GET and PUT operations.

For help with this field, ask your network administrator.

## Community embedded SMTP fields

The following are the maintenance fields for a community embedded SMTP transport server.

## Change this embedded transport server

Community: Worldwide

Server type: SMTP

Name:**\***

Host:

Port:**\***

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 these fields.

Host used by partners:**\***

Port used by partners:**\***

**Save changes** **Cancel**

Figure 31. Community embedded SMTP server settings tab

## Settings tab

### 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 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.

## 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	<b>server1.domain.com:4080</b>
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	<b>edi.domain.com:5080</b>
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	<b>edi.domain.com:5080</b>
Partner request	POST <b>/exchange/routingID</b> 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	<b>proxy.domain.com:8080</b>
Partner request	POST <b>http://edi.domain.com:5080/inbound/stuff</b> HTTP/1.1 HOST <b>edi.domain.com:5080</b> <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 <b>edi.domain.com:1453</b>
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.

## Concepts

- ◆ [Four exchanges required](#) on page 202
- ◆ [Trading delivery exchanges](#) on page 203
- ◆ [Integration exchanges](#) on page 206
- ◆ [Delivery exchange wizard](#) on page 212
- ◆ [Adding transports](#) on page 211
- ◆ [Post-processing configuration details](#) on page 265

## Procedure

- ◆ [Manage file system integration](#) on page 269
- ◆ [Set up default file system integration](#) on page 269
- ◆ [Enable, disable, test exchanges](#) on page 274
- ◆ [Set delivery exchange preferences](#) on page 276

## Pages and fields

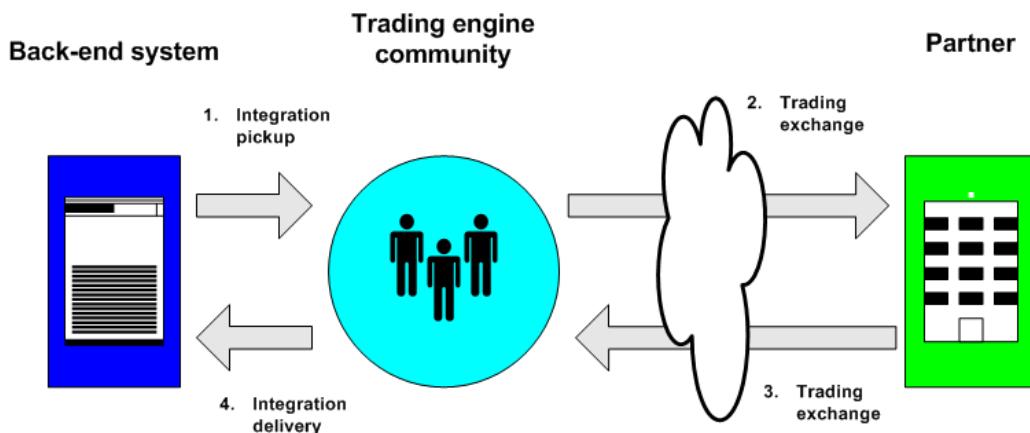
- ◆ [Detached e-mail for community](#) on page 213
- ◆ [Embedded e-mail for community](#) on page 215
- ◆ [E-mail transport for partners](#) on page 216
- ◆ [Embedded HTTP transport](#) on page 219
- ◆ [Staged HTTP](#) on page 277
- ◆ [HTTP transport for partners](#) on page 221
- ◆ [File system transport](#) on page 223
- ◆ [External FTP transport](#) on page 226
- ◆ [Embedded FTP transport](#) on page 230
- ◆ [SFTP transport](#) on page 240
- ◆ [JMS transport](#) on page 245
- ◆ [IBM MQSeries transport](#) on page 253
- ◆ [Web services API integration](#) on page 255
- ◆ [Web services message protocol](#) on page 257

# 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 32 illustrates this concept. The numbers in the diagram correspond to the preceding numbers.



**Figure 32. 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 or transports, including trading exchanges and pickup and delivery integration exchanges.

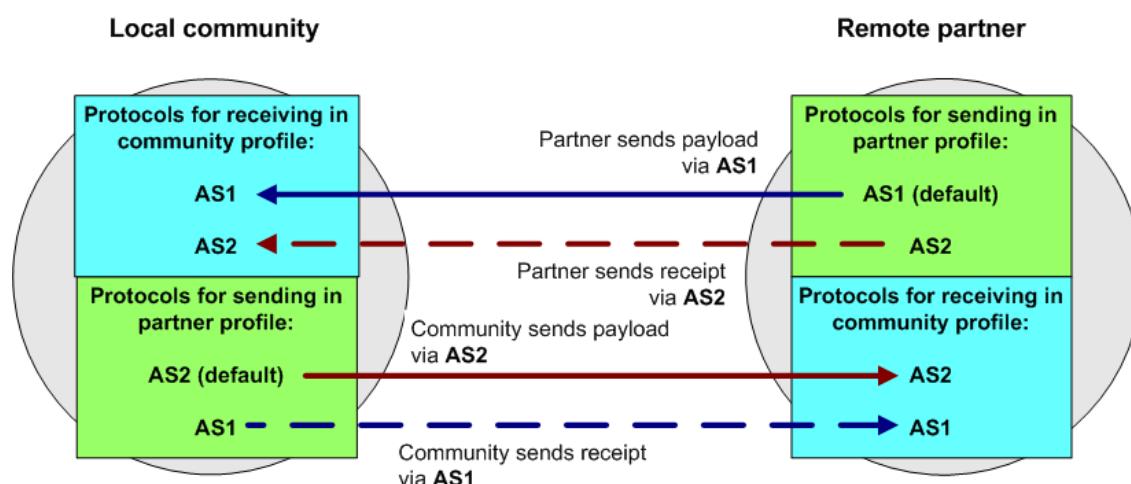
# Trading delivery exchanges

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.

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 33 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.



**Figure 33. Community and partner send via different message protocols**

Notice in Figure 33 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 in the partner profile for sending payloads. See [View settings between partners](#) on page 448.

Table 21 lists 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 206.

The packaging options column in Table 21 indicates whether a message protocol supports signing and encryption of messages using digital certificates. It also indicates whether message compression is supported.

**Table 21 - 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 Embedded FTP SFTP	Signing Encryption Compression
Secure file	File system FTP Embedded FTP SFTP Embedded HTTP Embedded HTTPS JMS MQSeries Staged HTTP web servlet	Signing Encryption Compression
Generic e-mail (formerly Secure e-mail)	SMTP/POP SMTP	Signing Encryption

---

**Table 21 - Supported trading exchanges**

<b>Message protocol</b>	<b>Transport options</b>	<b>Packaging options</b>
ebXML	Embedded HTTP Embedded HTTPS Staged HTTP web servlet Embedded SMTP SMTP/POP	Signing Encryption
ebXML intermediary	Embedded HTTP Embedded HTTPS Staged HTTP web servlet Embedded SMTP SMTP/POP	Not applicable. This protocol only re-routes messages. See <a href="#">Sending ebXML via an intermediary</a> on page 560.
Web services	Embedded HTTP Embedded HTTPS	Signing Encryption
RosettaNet 1.1  (requires special license)	Embedded HTTP Embedded HTTPS Staged HTTP web servlet	Signing Compression
RosettaNet 2.0  (requires special license)	Embedded HTTP Embedded HTTPS Staged HTTP web servlet	Signing Encryption Compression
No packaging  (formerly Other)	HTTP File system FTP Embedded FTP SFTP JMS MQSeries	None; messages are not signed, encrypted, or compressed

**Note:** The Synchrony Gateway HTTP interface transport available under the no packaging message protocol is for use only when integrating with Synchrony Gateway. See [Integrate with Synchrony Gateway](#) on page 663.

## If your partner uses Synchrony Gateway

If a partner uses Synchrony Gateway and your community wants to send messages to the partner via secure file message protocol HTTP, see [Receiver is Synchrony Gateway](#) on page 312 for important configuration information.

## 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](http://www.ebxml.org) or [www.oasis-open.org](http://www.oasis-open.org)

RosettaNet      [www.rosettanet.org](http://www.rosettanet.org)

## Interoperability

The trading engine has been certified for interoperability for AS1, AS2, AS3 and ebXML. See [www.ebusinessready.org](http://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 22 lists the available transports for integration pickup and delivery exchanges.

**Table 22 - Supported integration exchanges**

Transport	Integration pickup	Integration delivery
File system transport	✓	✓
File system with message meta-data (see <a href="#">File system transport</a> )		✓

**Table 22 - Supported integration exchanges**

Transport	Integration pickup	Integration delivery
External FTP transport	✓	✓
Embedded FTP transport	✓	✓
IBM MQSeries transport	✓	✓
JMS transport	✓	✓
SFTP transport	✓	✓
Web services API client (see <a href="#">Web services API integration</a> on page 255)		✓
Web services API server (see <a href="#">Web services API integration</a> on page 255)	✓	
Synchrony Integrator (see <a href="#">Integrate with Synchrony Integrator</a> on page 661)	✓	✓

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 541 or [RosettaNet support](#) on page 585.

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 362, [Message attributes tab](#) on page 364 and [Delivery criteria tab](#) on page 373.

**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 362.

## ***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 373.

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 269).

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 34 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: Acme

After this community endpoint has received and picked up messages, the following protocols are used to route messages to back-end systems.

<input type="checkbox"/>	Type	Location	Name	Delivery criteria	Test			Delete
<input type="checkbox"/>	Other (Plain text) / File system	C:\data\ediin		( Content MIME type equals application/EDI-X12 ) OR ( Content MIME type equals application/EDIFACT ) OR ( Content MIME type equals application/EDI-consent ) OR deliver by default	<a href="#">Test</a>			<a href="#">Delete</a>
<input type="checkbox"/>	Other (Plain text) / File system	C:\data\xmlin		Content MIME type equals application/xml	<a href="#">Test</a>			<a href="#">Delete</a>
<input type="checkbox"/>	Other (Plain text) / File system	C:\data\binaryin		Content MIME type equals application/octet-stream	<a href="#">Test</a>			<a href="#">Delete</a>

**Figure 34. 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 35 shows that a file system exchange is the default to use when inbound messages do not meet the criteria specified by the other exchanges.

## Pick an integration delivery exchange

### Community: Acme

After this community endpoint has received and picked up messages, the following protocols are used to route messages to back-end systems.

<input type="checkbox"/>	Type	Location	Name	Delivery criteria	Test			Delete
<input type="checkbox"/>	Other (Plain text) / File system	C:\data\other		Deliver by default	<a href="#">Test</a>			<a href="#">Delete</a>
<input type="checkbox"/>	Other (Plain text) / File system	C:\data\ediin		( Content MIME type equals application/EDI-X12 ) OR ( Content MIME type equals application/EDIFACT ) OR ( Content MIME type equals application/EDI-consent )	<a href="#">Test</a>			<a href="#">Delete</a>
<input type="checkbox"/>	Other (Plain text) / File system	C:\data\xmlin		Content MIME type equals application/xml	<a href="#">Test</a>			<a href="#">Delete</a>
<input type="checkbox"/>	Other (Plain text) / File system	C:\data\binaryin		Content MIME type equals application/octet-stream	<a href="#">Test</a>			<a href="#">Delete</a>

**Figure 35. Delivery exchange with no criteria at top of list is the default**

## ***Simulate sending messages to integration***

The user interface has a simulator for testing whether messages are being routed to the correct integration delivery exchanges. The simulator lets you pretend a community has received a message from a partner via a particular message protocol. You click a test button to see which exchange the community uses to send the “message” to integration.

If you have configured delivery criteria for an integration exchange, the simulator lets you test the configuration. For information about delivery criteria, see [Message attributes tab](#) on page 364 and [Delivery criteria tab](#) on page 373.

To open the message simulator, select **Trading configuration > Message simulator**. There also is a link for opening the simulator page at the bottom of the page that lists the integration delivery exchanges for a community. To open the exchange list, click **Integration delivery** on the navigation graphic on a community summary page.

Before using the simulator, you need at least one community profile and one integration delivery exchange for the community. You do not have to add an exchange for the community to receive messages from partners to use the simulator.

**Simulate a message to integration** [Close](#) [Help](#)

Use this simulator to test whether a community's integration delivery exchanges are working as intended.

Select a community and the protocol of a simulated message from a partner. Optionally, add metadata elements and values for integration delivery criteria. Run the test to see which delivery exchange is used.

Simulate inbound message for:

Simulate receiving message via:

Integration delivery criteria:  equals

Specified metadata

None

**Test results:**  
No tests have been run...

**Figure 36. Page to simulate sending messages to integration**

The following describes the fields on the simulator page.

### **Simulate inbound message for**

The community to receive the simulated message from a partner.

### Simulate receiving message via

The message protocol the community pretends to use to receive the message from the partner. You can select any protocol, regardless whether it has been configured in the community profile.

### Integration delivery criteria

Optional meta-data and values specifying delivery criteria for an integration exchange. Select a meta-data element, type a value and click **Add**. You can add multiple elements and values.

Click **Run test** to display test results. Click **Edit this exchange point** below the test results to change delivery criteria or other configuration.

The messages produced by the simulator are not real and are used only for testing. Message Tracker does not keep records of simulated messages.

## Adding transports

The following is a list of 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 adding a transport using the delivery exchange wizard.

- [Detached e-mail for community](#) on page 213
- [Embedded e-mail for community](#) on page 215
- [E-mail transport for partners](#) on page 216
- [Embedded HTTP transport](#) on page 219
- [Staged HTTP](#) on page 277
- [HTTP transport for partners](#) on page 221
- [File system transport](#) on page 223
- [External FTP transport](#) on page 226
- [Embedded FTP transport](#) on page 230
- [JMS transport](#) on page 245
- [IBM MQSeries transport](#) on page 253
- [Web services API integration](#) on page 255
- [Web services message protocol](#) on page 257

**Note:** The Synchrony Gateway HTTP interface transport available for a community under the no packaging message protocol is for use only when integrating with Synchrony Gateway. See [Integrate with Synchrony Gateway](#) on page 663.

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 203.

## 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.

When you add a transport with the wizard, you optionally can type a name for the exchange point. If you have many delivery exchanges, naming the transports you configure can help you to identify and manage the various exchanges. After adding an exchange, you can change the name on the transport's maintenance page. See [Transport maintenance](#) on page 295.

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.

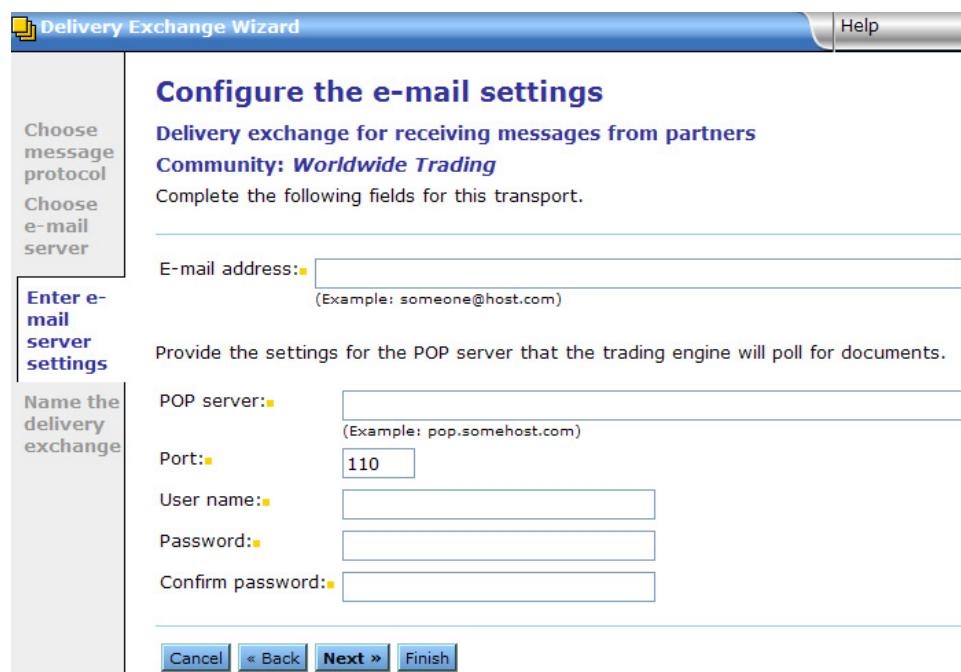
- While configuring a partner, the following link appears on the partner summary page, prompting you to complete this delivery exchange task. Once the task is completed, the prompt disappears.  
**Set up a delivery exchange for sending messages to this partner**
- Click **Add a delivery exchange** at 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 215. The delivery exchange wizard provides options for using a detached or embedded SMTP server.



**Figure 37. 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:

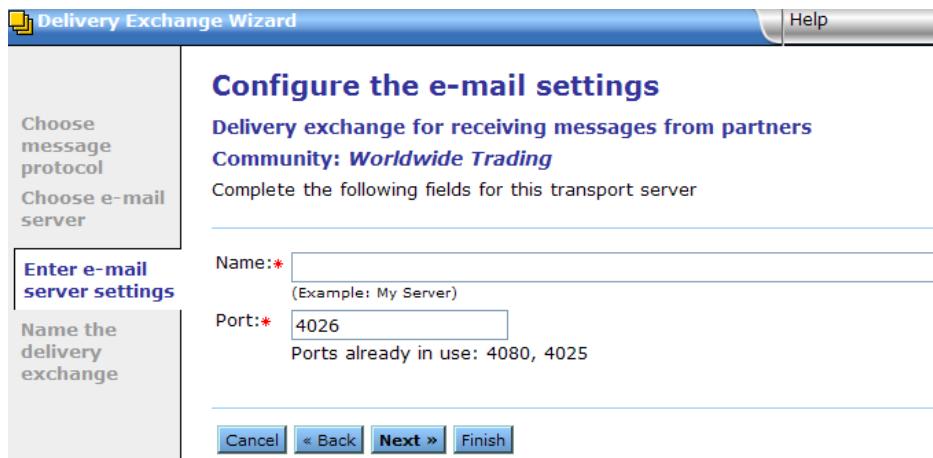
- ♦ Use the system's global embedded SMTP server
- ♦ Use a previously defined embedded SMTP server (if available)
- ♦ Define a new embedded SMTP server

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 185 or [Community embedded SMTP fields](#) on page 192.



**Figure 38. 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.

#### 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.



**Figure 39. Configure partner e-mail (1 of 2)**

The following three fields are shown in Figure 39.

### 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.

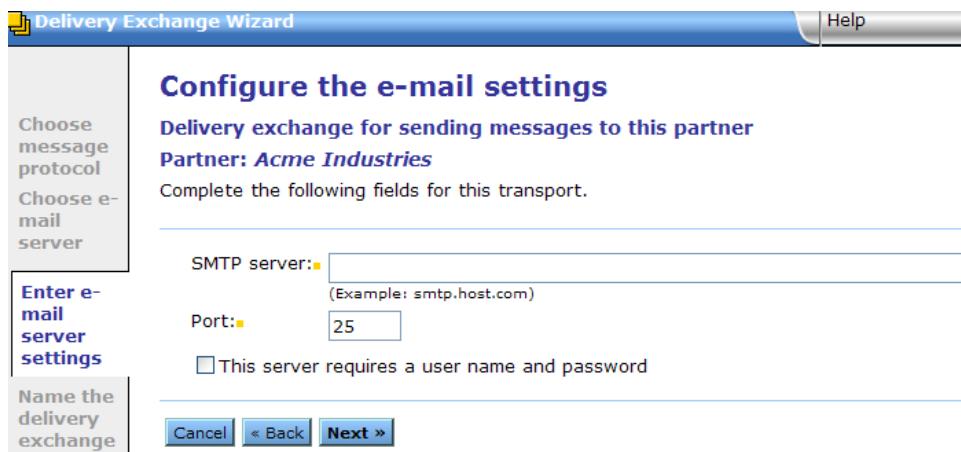
If you select this, click **Next**. The configuration is completed, unless you elect to type a name for the exchange. Click **Finish**.

See [Configuring external SMTP server](#) on page 23.

### 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 40.



**Figure 40. 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.

Click **Next**. The configuration is completed, unless you elect to type a name for the exchange. Click **Finish**.

# 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 HTTP server, the wizard uses the default routing ID for the community as the last string in the URL. You can accept or change the string. 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 embedded HTTPS server, you must add an SSL certificate for the server. After setting up the server in the delivery exchange wizard, go to the community profile summary page and click **Change an embedded transport server** near the bottom of the page. Click the name of the server to open the maintenance page. 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 183 for information about changing the configurations of embedded servers.



**Figure 41. Add embedded HTTP server in delivery exchange wizard (1 of 2)**

See [Transport guidelines](#) on page 637 for security guidelines for the embedded HTTP server.

The following fields are used in the delivery exchange wizard for adding an embedded HTTP server transport by defining a new embedded HTTP server.

#### 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 42.



**Figure 42. Add embedded HTTP server in delivery exchange wizard (2 of 2)**

### URL

The wizard displays the URL for the transport. This is the URL your partner uses to send messages to the community.

The last item in the URL is the community profile routing ID. This is a suggested value. You can accept it or type another string.

Click **Next** if you want to name the exchange. Otherwise, click **Finish**.

## 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 277.

## HTTP transport for partners

Communities can send messages to partners via an external HTTP server.



**Figure 43. 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 not encrypted.

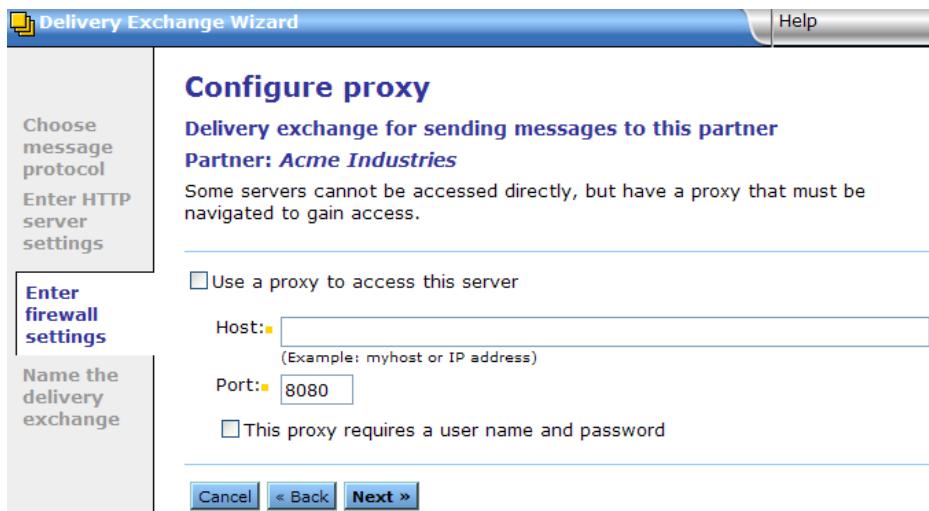
#### *Enable host name verification*

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 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 44.



**Figure 44. 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.

Click **Next**. Typing a name for the exchange is optional. Click **Finish**.

## File system transport

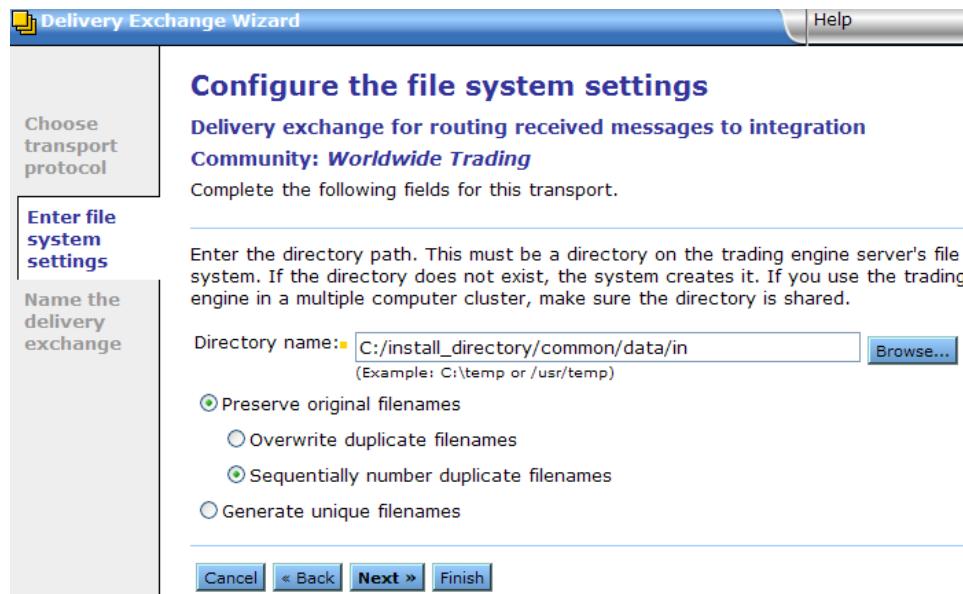
The file system transport can be used as a trading or integration exchange, but most often is used for integration.

**Note:** There is a quick way to set up directories for file system integration pickup and delivery. See [Manage file system integration](#) on page 269.

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.



**Figure 45. 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. When adding a file system transport, the delivery exchange wizard suggests a 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 the ebXML, RosettaNet or web services message protocols.

#### **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.

For most transports, the appended number is consecutively numbered. For FTP and SFTP, however, the appended number is hexadecimal and looks this: filename\_c4.

#### **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:

dabeeed45\_4cb.edi  
z47e4120\_4ce

Click **Next** if you want to name the exchange. Otherwise, click **Finish**.

## External FTP transport

The FTP transport can be used as a trading or integration transport.

**Note:** This topic describes configuring an external FTP server. For an embedded FTP server, see [Embedded FTP transport](#) on page 230.

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.

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 Interchange or Activator that does not support AS3.

---

**CAUTION:** 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.

---

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.SimpleDebug=info
```

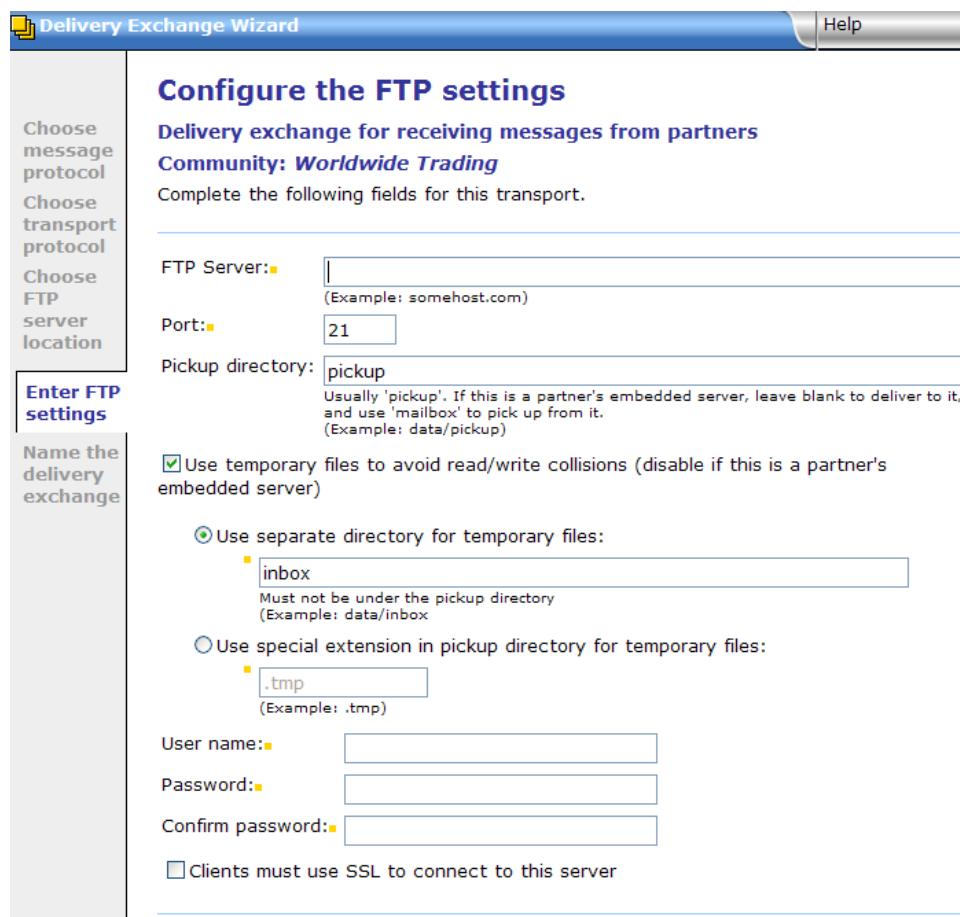
The debug setting logs each meta-command (for example, Send), followed by each primitive command as it is sent to the server (Rest, Stor, Rnfr, Rnto, and so on). Also, a message is logged each time a data connection is initiated or accepted. Each file name received during List operations is logged.

For more information see [System logs](#) on page 94.

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 525.



**Figure 46. 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 (embedded FTP default is 4021).

### 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.

If connecting to a partner's Activator embedded FTP server, use **mailbox** if picking up. Leave the field blank if delivering.

### **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.

If connecting to a partner's Activator embedded FTP server, clear the check box.

### ***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 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 not encrypted.

***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.

Click **Next** if you want to name the exchange. Otherwise, click **Finish**.

## Embedded FTP transport

A community can use an embedded FTP server to receive messages from partners or as an integration transport.

When you elect to set up an embedded FTP transport for a community, the delivery exchange wizard asks whether you want to:

- Use a previously defined embedded FTP server (if available)
- Define a new embedded FTP server

If you choose to use a previously defined embedded FTP server, the wizard prompts for the user account name. You can choose an existing account or define a new one. If you choose an existing account, you must choose a subdirectory within that account's home directory that is not assigned to any other exchange point.

If you choose to define a new embedded FTP server, the wizard prompts for a server name and port number. The wizard then asks for a user name and password for the new server. If the default selections already are in use, you must use another user name and password.

**Note:** For a description of configuring an external FTP server, see [External FTP transport](#) on page 226.

## **Types of embedded FTP servers**

There are two types of embedded FTP servers:

- 1 Trading servers** are used for receiving messages from partners. In the simplest case, a partner's FTP client connects to your embedded server to PUT a message for your community to pick up. There also is a more complex configuration — hosted embedded FTP — where your community lets partners connect to your embedded server to GET messages.
- 2 Integration servers** are used for retrieving messages from, or delivering messages to, back-end systems.

The delivery exchange wizard enforces the usage context for embedded FTP servers. For example, the wizard will not let you define a new embedded FTP trading server when the usage requires an integration server.

Configuring hosted partner accounts for embedded FTP servers requires a certain license permission. Your license.xml file must enable the following: `hostedPartnerFtpAccount`. Without this permission you can configure a partner profile to send messages via an external FTP server, but not an embedded server. This permission does not affect integration accounts; there is no restriction on adding hosted accounts from which a back-end system can pick up messages.

## FTP user accounts

When adding a delivery exchange using an embedded FTP server, you must specify an FTP user account. You can choose an existing account or define a new one. The trading engine internally associates user accounts with home directories for sending (PUT) or retrieving (GET) messages. There are three types of accounts.

Account type	Usage
Community FTP accounts	Community exchanges, trading servers only
Partner FTP accounts	Community and partner exchanges, trading servers only
Integration FTP accounts	Community exchanges, integration servers only

User names are shared by all trading servers. This enables a load balancer to send a request to any trading server. However, user names associated with integration servers are not related to user names shared by trading servers. For example, the user name **user1** on the trading side is completely different from **user1** on the integration side.

The following topics describe each account type.

### Community FTP accounts

Community FTP accounts are associated only with community delivery exchanges for receiving messages from partners.

Partners can perform PUT operations to community accounts associated with trading exchanges, but not GET operations. This is to prevent partners from accessing each others' files.

Partners can drop packaged messages directly into the home directory for a community account. As a result, community FTP accounts can be referred to as shared accounts.

When a community profile is exported as a partner profile file to be imported by partners, the community FTP account is exported with the delivery exchange.

To avoid file name collisions you can use the [Message attributes tab](#) on the delivery exchange to specify subdirectories where partners can place files based on the sender routing ID. This also allows identifying the sender

when binary files are dropped off. For example, the following subdirectory scheme could be used for two partners (p1 and p2) to drop off files for community c1:

Subdirectory	Purpose
p1/c1	p1 sends to c1
p2/c1	p2 sends to c1

If a partner-specific subdirectory scheme is used, the partner must manually specify the subdirectory after importing the community's profile. If there is only one community and it has only one routing ID, the second directory level is unnecessary.

## Partner FTP accounts

Partner FTP accounts are used only for trading. The accounts typically are used for outbound trading via delivery exchanges set up in partner profiles. But the same accounts can be used for inbound trading via delivery exchanges set up in community profiles. In this case they can be used in lieu of, or in addition to, community accounts.

The user interface segregates the two purposes. When adding a partner embedded FTP delivery exchange, the wizard suggests the **/mailbox** subdirectory under the home directory. This is where the trading engine will deliver messages for the partner to pick up. This can be thought of as a *hosted* partner exchange, since the partner is picking up messages from your server. If the same partner account is added to an existing community delivery exchange, the system will suggest the home directory (/). This ensures outbound and inbound messages are not confused.

For hosted partner exchanges, you must inform the partner performing the GET operation of the FTP host name or IP address, the port number, the path, and the server user name and password. If your partner uses Interchange or Activator, the partner must add a community delivery exchange for receiving messages via an external FTP server (see [External FTP transport](#) on page 226 for configuration information). The default value of pickup for the pickup directory must be changed to **mailbox** or whatever directory you specified. Also, clear the check box for **use temporary files**.

Partner FTP accounts cannot be used by integration exchanges.

### Outbound trading example

In this example, an administrator adds an outbound hosted FTP delivery exchange for **partner1**. The administrator associates the user account **p1**. The user interface suggests **/mailbox** as the subdirectory where the partner will GET messages.

Optionally, for binary trading the [Message attributes tab](#) can specify subdirectories where the trading engine will deliver files based on the routing ID of the sending community. For instance:

Subdirectory	Purpose
p1/mailbox/c1	p1 GETS messages from community1
p1/mailbox/c2	p1 GETS messages from community2

Since partner FTP accounts are partner-specific, message attribute mapping only is needed for identifying the community if there is more than one possible sending community or there is no other way to identify the community. This would be the case when trading binary files, which have no packaging or internal identifying information.

Message attribute mapping also can be used to sort messages into arbitrary subdirectories based on other meta-data besides the sender or receiver. For example, an inline process could assign meta-data items to outbound messages that would cause them to be sorted into subdirectories based on mappings specified on the message attributes tab.

### Inbound trading example

This example is for traders who do not want to upload files to a shared community FTP account.

An administrator adds or selects a community embedded FTP delivery exchange. The exchange can be one already associated with a community FTP account like c1. The exchange provides an anchor for one or more partner FTP accounts where files may be dropped off for the community.

The administrator selects the FTP users tab on the community delivery exchange. There the administrator can add a new partner account or select an existing one.

The user interface suggests **/** as the home directory for the FTP account where partners can drop files for the community. In contrast, **/mailbox** is the suggested directory when a hosted delivery exchange is added for a partner to GET messages.

**Note:** Embedded FTP servers treat paths beginning with a slash as starting at the home directory for the user account currently logged in.

## Integration FTP accounts

Integration FTP accounts are associated only with integration delivery exchanges for retrieving messages from, or delivering messages to, backend systems.

Integration pickup exchanges are not tied to a particular community. When defining for the first time an embedded FTP pickup for integration, the delivery exchange wizard suggests the generic name **integration** as the FTP account name and password. The same user name is offered for the first integration delivery exchange. The difference is \out is suggested as the pickup subdirectory name and \in is suggested as the delivery subdirectory name. You can change the user names and subdirectories as long as the same combination of user name and subdirectory is not specified for two exchanges. This is true for all embedded FTP exchanges.

## ***Firewalls, load balancers and embedded FTP***

FTP servers present challenges with firewalls and load balancers. If you have a corporate firewall and the computer running the trading engine also has a Windows firewall, turn off the Windows firewall. Ask the network administrator to open the control port for your embedded FTP server (default is 4021) on the network firewall. You also may need to ask the administrator to open a range of ports for passive connections (see [Allowed passive ports](#) on page 192).

In the early days of the Internet it was common for an FTP client to use port mode. The data connection was established from the server back to the client, even though the original control connection was always established from the client to the server.

But usually the client is less prepared than the server to open firewall ports, and passive mode has become the standard (Windows command line FTP client is an exception). In passive mode when the client wants to perform a GET, PUT or LIST operation, the server creates a temporary socket listening on a port specifically for that client, usually at a high number like 50,000. The client then makes the data connection to the server, performs the transfer and the connection is dropped. The control connection remains active.

Most firewalls in use today allow the client through on the temporary data port, even if the port is not explicitly configured to be open. This is safe because the firewall can eavesdrop on the initial control connection from

the client and notice it is an FTP connection. If the connection is non-SSL, the firewall also can notice the temporary port the server assigned for the client to establish the data connection. Subsequently when the client tries to connect again on that port, the firewall allows the connection on a one-time basis to the temporary port.

Load balancers present a similar problem. If the load balancer is FTP-aware, it can route incoming data connections to the same server as the associated control connection. But the load balancer cannot do this in the case of SSL. It must be configured to route all connections from the same origination IP address to the same server until there are no more connections from that IP to that server.

## ***Embedded FTP fields***

The following fields are used in the delivery exchange wizard for adding an embedded FTP server transport for a community by defining a new embedded server.



**Figure 47. Add embedded FTP server in delivery exchange wizard (1 of 3)**

### **Name**

Type a name for the new embedded FTP server. This can be any name you want. You can select this server when setting up additional embedded FTP delivery exchanges later.

If this is the first server, the wizard suggests the name **Trading FTP** if the server is to be used for receiving messages from partners or **Integration FTP** if the server is to be used for exchanging messages with a back-end system.

## Port

The port number that listens for incoming FTP connections. This is known as the control port. The wizard suggests a port not in use and displays a list of ports in use by the trading engine. You can use the suggested port or another.

If this is the first server, the wizard suggests — if not already in use — 4021 for a trading server or 5021 for an integration server. For security reasons, trading and integration servers must use different ports.

Click **Next** to continue the configuration. See Figure 48.



**Figure 48. Add embedded FTP server in delivery exchange wizard (2 of 3)**

You have these options for selecting a user account:

- ◆ Select an existing account from the drop-down list
- ◆ Define a new account
- ◆ Define an account later (see [Embedded FTP maintenance](#) on page 325)

If you elect to define a new account, user name and password fields appear. If you are defining the first user account, the wizard suggests for the user name and password the community default routing ID for a trading server or **integration** for an integration server.

#### **User name**

The user name to connect to the server.

Not only does your partner use this name to connect, but the trading engine creates a directory of the same name. This is the home directory for the FTP account. It is where a partner will send messages to your community via FTP. If you use the default location for the common directory, the directory is at [install directory]\common\data\ftp\users\trading. But if you are configuring an integration transport, the path is [install directory]\common\data\ftp\users\integration.

---

**CAUTION:** Do not configure a back-end system to retrieve files from or write files to common\data\ftp\users\trading. Doing so could result in operational difficulties. The back-end system always should interact with transports defined for integration and allow the trading engine to route messages to and from trading transports.

---

#### **Password**

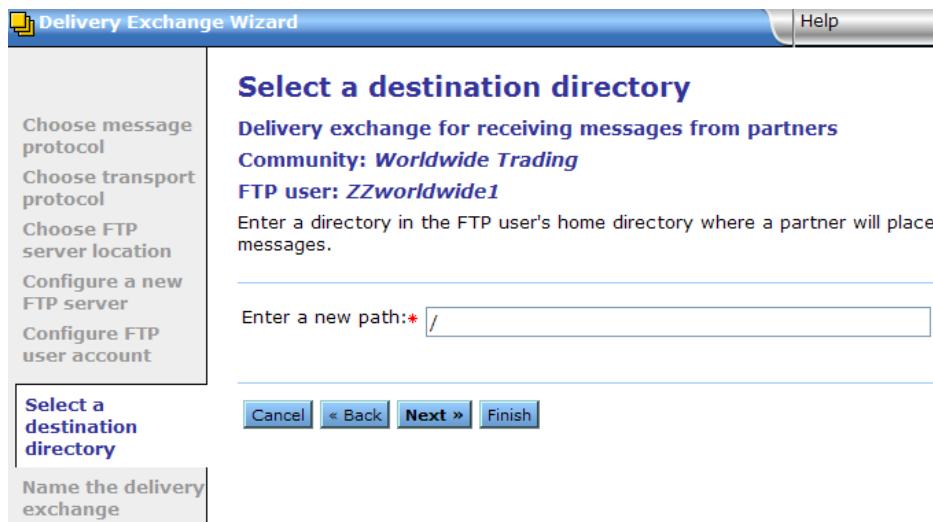
The password to connect to the server.

#### **Confirm password**

The password to connect to the server.

Anonymous connections are not supported.

Click **Next** to continue the configuration. See Figure 49.



**Figure 49. Add embedded FTP server in delivery exchange wizard (3 of 3)**

### Enter a new path

This field enables you to associate the same embedded server and user account to multiple subdirectories. Each subdirectory can be used by a single partner to send messages to your community via FTP.

If you leave this field blank, the effective directory where a partner will upload messages is the top-level directory. For example:

common\data\ftp\users\trading\[user account]

But if you add an amended path, the effective directory is the specified subdirectory. For example:

common\data\ftp\users\trading\[user account]\[amended path]

For the first integration server, the wizard suggests an amended path of **in** for integration delivery or **out** for integration pickup. By default the home directory (/) is not used, but you can change this.

The trading engine keeps track of the embedded FTP directory structure for you. Do not manually change any directories the trading engine creates for managing messages transported via embedded FTP servers.

Click **Next** if you want to name the exchange. Otherwise, click **Finish**.

## 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.

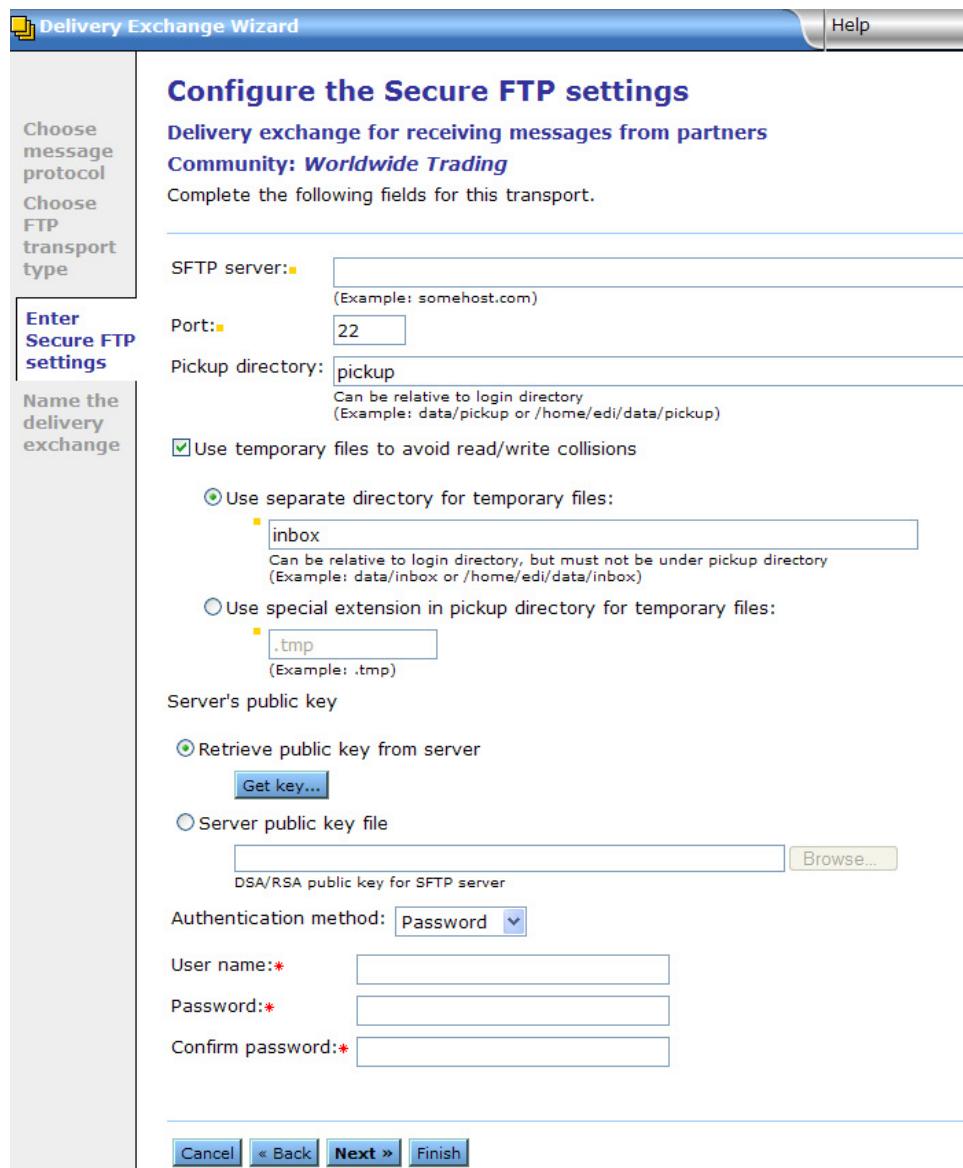
The trading engine supports limited SFTP functionality as the following notes:

- ◆ Only supports SSH 2.0.
- ◆ 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.

For more information about SSH see:

- ◆ <http://www.ietf.org/html.charters/secsh-charter.html>
- ◆ <http://www.openssh.com/txt/draft-ietf-secsh-filexfer-02.txt>
- ◆ <http://www.openssh.com/faq.html>



**Figure 50. 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 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.

#### **Server's public key**

You have two options for designating the RSA or DSA public key for the SFTP server. The trading engine uses the key to authenticate the server.

##### ***Retrieve public key from server***

Click **Get Key** to have the trading engine retrieve the public key for the SFTP server. The server name and port number entered on this page must be correct for this option to work.

##### ***Server public key file***

Type the path to the file containing the public key for the SFTP server or click **Browse** to locate the file. You may have to ask the server administrator for the file path.

#### **Authentication method**

There are three methods.

**Password** requires entering the user name and password for connecting to the server. The user name and password are sent over an encrypted connection to authenticate the user to the server. Although this option offers ease of administration, the password is vulnerable because it is sent every time a connection is made. The password could be compromised if the server is ever compromised.

**Hostbased** requires entering the user name of the server, the path to the private key and the password for the private key, if any. Authentication relies on the private key on the client side and the public key on the server side. With this option, one public-private key pair is shared for all users that connect from a single client. Care is advised to make sure a regular user does not have access to the private key.

**Public key** requires entering the user name of the server and the path to the private key. A password for the private key is optional but recommended. Authentication relies on the private key on the client side and the public key on the server side. Public key authentication requires administrators to create public-private key pairs for each user and then place the public key on the host in each user's home directory.

Click **Next** if you want to name the exchange. Otherwise, click **Finish**.

## 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, DELETED, 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.

**Note:** This information for setting up JMS exchanges also applies to the Synchrony Integrator exchange, a special JMS transport that enables integration between Activator and the Synchrony Integrator message translator. If you are integrating with Integrator, see [Integrate with Synchrony Integrator](#) on page 661 for more information.

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 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 Activator (for example, t3://node1:7001,node2:7002,node3:7003). At runtime, the JMS client round-robs 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 directory containing 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. Do not put the JAR in the application's corelib directory.

In Windows, add the JAR path at the end of the class path in the following lines in the environment file:

```
set CYC_CP=%CYC_CP%;..\corelib\db\sqlserver;..\corelib\db\oracle;..\corelib\db\db2;..\corelib\db\derby;..\corelib\db\mysql
```

For example, for Apache ActiveMQ on Windows, the entry would be similar to the following:

```
set CYC_CP=%CYC_CP%;..\corelib\db\sqlserver;..\corelib\db\oracle;..\corelib\db\db2;..\corelib\db\derby;..\corelib\db\mysql;..C:\apache-activemq-4.1.1
```

In this example, the path is to the directory containing the JAR file, but not the path to the JAR itself. This is recommended.

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 GatewayInterchangeService.ini file, which also is in the bin directory.

```
CYC_CP=..\classes;..\site\conf\resources;..\site\jars;..\conf\resources;..\jars;..\corelib;..\corelib\db\sqlserver;..\corelib\db\oracle;..\corelib\db\db2;..\corelib\db\derby;..\corelib\db\mysql
```

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}:${CYC_DIR}/site/conf/resources"
CYC_CP="${CYC_CP}:${CYC_DIR}/site/jars"
CYC_CP="${CYC_CP}:${CYC_DIR}/conf/resources"
CYC_CP="${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}:${CYC_DIR}/corelib/db/derby"
CYC_CP="${CYC_CP}:${CYC_DIR}/corelib/db/mysql"
```

After the last line, insert a line specifying the JAR location. For example:

```
CYC_CP="${CYC_CP}:/apache-activemq-4.1.1"
```

## 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.

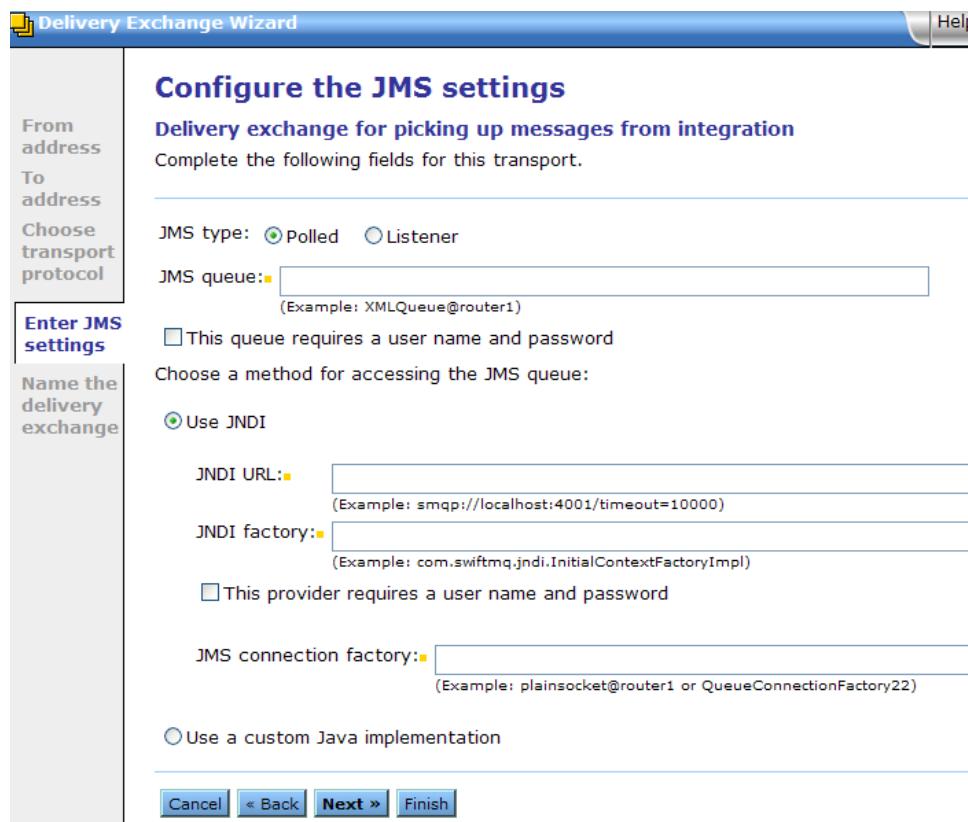
Add a message queue on Oracle AQ. The queue payload type must be one of the following:

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 `rdbms/jlib` directory. Paste the files to the 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.



**Figure 51. 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. (If Sybase EAServer is the JMS provider, you must select **Polled**.)

**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. (If you use WebLogic, the Allow Close In On Message check box for the queue factory must be selected in the WebLogic user interface.)

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@routename** 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.

**Database name.** The name of the database that contains the 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**.

#### **Send payload via file system**

Select this check box to have payloads sent by a file system. You can specify the size of payloads to send and the path for payload files. The receiver uses the path to retrieve the files.

This option is available when the exchange is for integration delivery and sending to partners.

Click **Next** if you want to name the exchange. Otherwise, click **Finish**.

## **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):
```

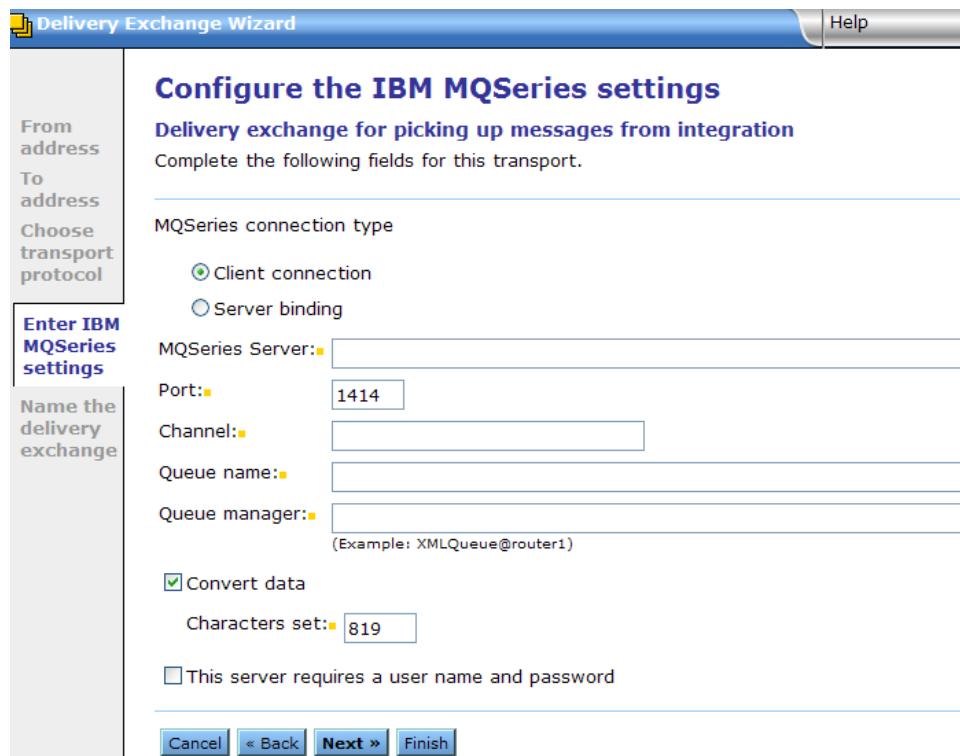
After prompting for the initial configuration information, you can use one of the testing commands, which are:

- (c)onnect
- (d)isconnect
- (s)end
- (r)etrieve
- (a)cnowledge

## 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.



**Figure 52. Add IBM MQSeries transport in delivery exchange wizard**

The following fields are used in the delivery exchange wizard for configuring this transport.

#### MQSeries connection type

Select **Client connection** to use a channel connection, on the local machine or via the network, to connect to a queue manager.

Select **Server binding** to use an API connection, via shared memory, to a local queue manager.

#### 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.

**Convert data**

Select to convert the characters set of messages received from a queue to the set specified in the Characters set field. Clear the check box to turn off data conversion. This setting does not apply to messages outbound to a queue.

**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**

If selected, type a user name and password to connect to the server.

Click **Next** if you want to name the exchange. Otherwise, click **Finish**.

# Web services API integration

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.

There are two ways to get the WSDL document that describes the requirements for both the server and client.

The first way is to set up a web services API server integration pickup exchange and then point a browser to the following URL:

**<http://localhost:5080/services/MessageService?wsdl>**

The variable **localhost** is the fully qualified domain name or IP address of the computer running the trading engine.

If you have the optional Software Development Kit, the second way is to get the following file in the SDK directory tree:

sdk\wsdl\TradingEngineNodeServices\MessageService.wsdl

Sample client implementations also are in the SDK directory tree. For more information see the SDK Developer's Guide.

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 53).



**Figure 53. Add API client transport in delivery exchange wizard**

Click **Next** if you want to name the exchange. Otherwise, click **Finish**.

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 515.

## Web services message protocol

Communities can use the web services message protocol to trade messages with partners. The web services framework allows sending and receiving SOAP messages with payloads included within the SOAP body or as MIME attachments. Included is support for WS-Security and WS-Addressing. In addition, custom SOAP handlers can be configured to perform processing on the SOAP messages. Possible processing operations include payload transformation, payload validation and WS\* header processing.

Architecturally, the web services framework is designed to pass messages through a chain of handlers. Each handler is responsible for one operation on a message. For example, WS-Security is implemented as a handler. For an inbound message the WS-Security handler is responsible for decrypting and signature validation. For an outbound message, the WS-Security handler is responsible for encryption and signature creation. Other handlers are responsible for other message operations. The handlers within a chain are run in order of configuration.

Adding a transport in the delivery exchange wizard for the web services message protocol is just like setting up an HTTP exchange. For information see [Embedded HTTP transport](#) on page 219 and [HTTP transport for partners](#) on page 221.

Also see [Web services trading maintenance](#) on page 358 for advanced tuning controls and [Web services default settings](#) on page 470 for information about collaboration settings.

## **Payload packaging**

The payload transferred in a SOAP message can be in the SOAP body or packaged as a MIME attachment.

When packaging an outbound message, a decision must be made on where to place the payload. The packaging decision is specified as part of the binary collaboration. By default outbound payloads are placed in the SOAP body of the packaged message. The payload also can be packaged as an attachment by changing the payload location field on the web services tab in the collaboration settings area of the user interface. Setting the payload location to **none** has the effect of not attaching the payload as an attachment or in the SOAP body. In the **none** case, a SOAP handler must be configured to attach the payload where desired. Doing this requires obtaining the optional Software Development Kit.

The payload location collaboration setting in the user interface setting can be overridden by setting the meta-data items named AxisShouldAddPayload and AxisAddPayloadAsAttachment.

For AxisShouldAddPayload the value should be **true** to add the payload to the body or as an attachment and **false** to not automatically include the payload.

The AxisAddPayloadAsAttachment value should be **true** to package payloads as attachments. The value should be **false** to package payloads as part of the SOAP body.

The SOAP body can only contain XML payloads. Other payload types and large XML payloads should be packaged as attachments.

## **Payload unpackaging**

An inbound SOAP message can have payloads in the SOAP body or as attachments.

The default behavior is to integrate payloads found in the SOAP body and integrate any MIME attachments. The Advanced tab for the web services delivery exchange for a community allows for SOAP body and attachment integration to be disabled. Also, the AxisShouldIntegrateSoapBody and AxisShouldIntegrateAttachments message meta-data values can be set to override the values on the Advanced tab.

## Message meta-data

The following defines the message meta-data elements.

### **AxisShouldAddPayload**

Can be used to override the value of the payload location field in the collaboration settings area of the user interface. Specifies whether to automatically add the payload to an outbound SOAP message. A value of **false** indicates the payload should not be added to the SOAP message. AxisAddPayloadAsAttachment specifies the location to add the payload if AxisShouldAddPayload is **true**.

### **AxisAddPayloadAsAttachment**

Can be used to override the value of the payload location field in the collaboration settings area of the user interface. Specifies whether to add the payload as an attachment or in the SOAP body.

### **AxisShouldIntegrateSoapBody**

Can be used to override the integrate SOAP body setting on the Advanced tab of the maintenance page for a community web services delivery exchange. Specifies whether to integrate the contents of the SOAP body.

### **AxisShouldIntegrateAttachments**

Can be used to override the integrate attachments setting on the Advanced tab of the maintenance page for a community web services delivery exchange. Specifies whether to integrate the attachments of the SOAP message. The value defaults to **true**.

### **AxisMessageType**

Specifies whether a message is a request or a response. This is set by the trading engine.

### **SOAPAction**

The SOAP action header value. If specified for outbound messages, the value is used when packaging. If no value is specified, SOAPAction defaults to a blank string.

### **ValidateBodyXML**

Specifies whether XML payloads should be schema-validated before adding the payload to the SOAP body of an outbound message. A value of **true** enables validation. Validation is **false** by default.

## **WS-Security handler**

Activator has a WS-Security handler to use for signing and encrypting SOAP messages and for validating signatures and decrypting. The WS-Security handler supports signature and encryption operations on the SOAP envelope and attachments.

Only the X.509 certificate profile is supported. The SAML profile, Username Token profile and any unmentioned security profiles are not supported.

The web services security settings in the user interface are implemented in the **WSSSenderHandler** and **WSSReceiverHandler**.

The collaboration settings area of the user interface allows configuration of signing and encryption parameters. See [Web services default settings](#) on page 470.

## **WS-Addressing handler**

Activator has a WS-Addressing handler for adding routing information to the SOAP header and interpreting routing information. The WS-Addressing handler supports the MessageId, From and To elements. Other elements are ignored.

## **Default configuration**

The default configuration of the web services message protocol enables trading of secure XML payloads between two instances of Activator. The default configuration is somewhat arbitrary, but useful to get two instances of Activator trading quickly. However, most users probably will want a different configuration. That requires obtaining the optional Software Development Kit.

## **Default packaging configuration**

Under the default configuration for packaging:

- ◆ Payload is placed in the SOAP body.
- ◆ WS-Security is enabled. The SOAP body is signed and encrypted.
- ◆ SOAP action value is blank.
- ◆ WS-Addressing header is added. Included are the sender and receiver routing IDs and a message ID.

## Default unpackaging configuration

Under the default configuration for unpackaging:

- ◆ SOAP body contents are integrated.
- ◆ Attachments are integrated.
- ◆ Synchronous acknowledgements are not enabled, so a 204 HTTP response is sent immediately on completion of unpackaging.
- ◆ WS-Addressing header is expected. Sender and receiver routing IDs and message ID are expected.

## Message meta-data documents

The trading engine supports message meta-data documents when file system integration is used with the web services message protocol. These XML documents are the interface between the trading engine and the back-end system.

There are two types of MMDs:

- 1** An MMD generated by a back-end system and retrieved from an integration pickup exchange. The MMD provides the location of a payload for the trading engine to pick up, package and send to a partner via the web services message protocol. The MMD also provides information the trading engine uses to process the outbound message.
- 2** An MMD the trading engine generates and sends to the back-end via an integration delivery exchange. The MMD contains information about a payload received from a partner via the web services message protocol. To trigger the trading engine to generate an MMD, you must set up a file system with message meta-data integration delivery exchange for your community.

The following topics describe each type of MMD.

## MMD from integration

Figure 54 is an example of an MMD from integration. The back-end system must generate an outbound MMD.

```
- <MessageMetadataDocument documentId="ID1234567890" protocol="webservices"
  protocolVersion="1.0">
  <Metadata name="From" type="string">Axway</Metadata>
  <Metadata name="To" type="string">IBM</Metadata>
  <Metadata name="MessageID" type="string">ID1234567@ws.com</Metadata>
  <Metadata name="AxisHandlerConfigWSDD">default_send_handler.wsdd</Metadata>
  <Metadata name="ws IsSyncResponse">false</Metadata>
  <Metadata name="ConnectionId">http1234567890</Metadata>
  <Metadata name="SOAPAction">WebServices</Metadata>
  <Metadata name="ValidateBodyXML">false</Metadata>
- <MessagePayloads>
  - <Payload id="0784247" packagingLocation="Attachment">
    <RemovePayloadAfterProcessing>false</RemovePayloadAfterProcessing>
    <MimeType>smallXmlPO</MimeType>
    <MimeType>application/xml</MimeType>
    <Location type="filePath">test/data/xml/smallxmlPO.xml</Location>
  </Payload>
  - <Payload id="0784248" packagingLocation="SOAPBody">
    <RemovePayloadAfterProcessing>false</RemovePayloadAfterProcessing>
    <MimeType>smallXmlPO1</MimeType>
    <MimeType>application/xml</MimeType>
    <Location type="filePath">test/data/xml/smallxmlPO1.xml</Location>
  </Payload>
</MessagePayloads>
</MessageMetadataDocument>
```

**Figure 54. Example of MMD from integration**

The following describes the MMD.

### protocol and protocolVersion

These attributes of MessageMetadataDocument element should be set to **webServices** and **1.0**, respectively. Required.

### From

This metadata item is the routing ID of the sending party.  
Required.

### To

This metadata item is the routing ID of the receiving party.  
Required.

### MessageId

This metadata item is a unique identifier for a message that conforms to [RFC 2822](#). Optional.

### **AxisHandlerConfigWSDD**

This metadata item specifies a Web Service Deployment Descriptor file. A WSDD file defines the SOAP handlers for the trading engine to invoke when packing or unpacking SOAP messages. Optional.

### **ws.IsSyncResponse**

This metadata item indicates whether the message is a synchronous response. Optional.

### **ConnectionId**

This metadata item specifies a unique identifier of the embedded HTTP server connection through which to send a synchronous response after receiving a message from a partner. Optional.

### **SOAPAction**

This metadata item specifies the SOAP action header value. If specified for an outbound message, the value is used when packaging. If no value is specified, SOAPAction defaults to a blank string. Optional.

### **ValidateBodyXML**

This metadata item specifies whether XML payloads should be schema-validated before adding the payload to the SOAP body of an outbound message. A value of **true** enables validation. Validation is **false** by default. Optional.

### **packagingLocation**

This optional attribute of the MessagePayloads element has the following valid values:

SOAPBody  
Attachment  
none

Only one payload can be attached to the SOAPBody, and only one SOAPBody value is allowed if multiple payloads are associated with the MMD.

The packagingLocation attribute replaces the attributes AxisShouldAddPayload and AxisAddPayloadAsAttachment. These two attributes should not be used in a web service MMD.

## MMD to integration

Figure 55 is an example of an MMD to integration. The trading engine generates an inbound MMD. After receiving a SOAP message from a partner, the trading engine unpacks the payload into a separate file.

```
- <MessageMetadataDocument id="ID124305781171657816783pnuechterlein-nc8430"
  protocol="webservices" protocolVersion="1.0">
  <Metadata name="From">g</Metadata>
  <Metadata name="To">nc8430</Metadata>
  - <Metadata name="MessageId">
    uuid:M1171657803697.4819@pnuechterlein-g_te5226705375118153788
  </Metadata>
  <Metadata name="SOAPAction">WebServices</Metadata>
  <Metadata name="AxisMessageType">request</Metadata>
  <Metadata name="ConnectionId">http1234567890</Metadata>
</MessageMetadataDocument>
```

**Figure 55. Example of MMD to integration**

The following describes the MMD.

### protocol and protocolVersion

These attributes of MessageMetadataDocument element should be set to **webServices** and **1.0**, respectively.

### From

This metadata item is the routing ID of the sending party.

### To

This metadata item is the routing ID of the receiving party.

### MessageId

This metadata item is a unique identifier for a message that conforms to [RFC 2822](#), if not empty.

### SOAPAction

This metadata item specifies the SOAP action header value, if not empty.

### AxisMessageType

This metadata item specifies whether a message is a request or a response.

### ConnectionId

If a synchronous business response is expected, this metadata item specifies an internal unique identifier of the embedded HTTP server connection.

## 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 path for the script file 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 path for the script file in the post-processing script field. Click **Save changes**.

When entering the path for a post-processing script in the user interface, we recommend using a relative path rather than an absolute path. Relative paths are preferred if you export partner profiles files for back-up purposes or clone partner profiles in a peer network environment.

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.
%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 266.

Add the meta-data elements you want to shellscriptrmetadata.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 shellscriptrmetadata.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 273
- [Post-processing to route inbound messages](#) on page 273

## ***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 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** Use the suggested top-level directory for the default directories or specify one (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 56 and Figure 57 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. All communities check all of the transports for outbound messages. But a community only picks up the messages that pertain to it and its partners.

Enabled exchanges		Disabled exchanges
Type	Location	Name
<input type="checkbox"/> Other (Plain text) from File system	C:\data\ediout	<input type="button" value="Test..."/> <input type="button" value="Delete"/>
<input type="checkbox"/> Other (Plain text) from File system	C:\data\xmlout	<input type="button" value="Test..."/> <input type="button" value="Delete"/>
<input type="checkbox"/> Other (Plain text) from File system	C:\data\binaryout	<input type="button" value="Test..."/> <input type="button" value="Delete"/>

Figure 56. Default integration pickup exchanges

### Pick an integration delivery exchange

Community: *Acme*

After this community endpoint has received and picked up messages, the following protocols are used to route messages to back-end systems.

Type	Location	Name	Delivery criteria	Test
<input type="checkbox"/> Other (Plain text) / File system	C:\data\ediin		( Content MIME type equals application/EDI-X12 ) OR ( Content MIME type equals application/EDIFACT ) OR ( Content MIME type equals application/EDI-consent ) OR deliver by default	<input type="button" value="Test"/> <input type="button" value="Up"/> <input type="button" value="Down"/> <input type="button" value="Delete"/>
<input type="checkbox"/> Other (Plain text) / File system	C:\data\xmlin		Content MIME type equals application/xml	<input type="button" value="Test"/> <input type="button" value="Up"/> <input type="button" value="Down"/> <input type="button" value="Delete"/>
<input type="checkbox"/> Other (Plain text) / File system	C:\data\binaryin		Content MIME type equals application/octet-stream	<input type="button" value="Test"/> <input type="button" value="Up"/> <input type="button" value="Down"/> <input type="button" value="Delete"/>

Figure 57. 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 365](#)

### **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.

BizTalk from and to XPaths are set up by default. If you need XPaths other than BizTalk, do the following:

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.

Make sure select **Always parse for the address** is selected. Also, make sure **If the document is XML, use XPaths to locate the address** is selected. Select a document type or click **XPath wizard** and point the wizard to the sender address in a sample XML document. Click **Finish** to close the wizard. On the from address tab, click **Save XPaths** and **Save changes**.

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 365.

## ***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 <b>Delivery exchange</b> .
Integration pickup	On the navigation graphic at the top of the community summary page, click <b>Integration pickup</b> .
Integration delivery	On the navigation graphic at the top of the community summary page, click <b>Integration delivery</b> .
Send messages to partners	On the navigation graphic at the top of the partner summary page, click <b>Delivery exchange</b> .

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. If there are multiple exchanges, the one at the top of the list is the default exchange. Use the up and down arrows to change the default exchange.

Type	<input type="checkbox"/>	Transport	Location	Name	Test		
EDIINT AS1	<input checked="" type="checkbox"/>	-Delivery & pickup: Embedded SMTP	ZZworldwide@PLorentz-T41.cyclonecommerce.com	email1	<input type="button" value="Test"/>	<input type="button" value="Up"/>	<input type="button" value="Down"/> <input type="button" value="Delete"/>
EDIINT AS2	<input type="checkbox"/>	-Delivery & pickup: Embedded HTTP	http://PLorentz-T41.cyclonecommerce.com:4080/exchange/ZZworldwide	http1	<input type="button" value="Test"/>	<input type="button" value="Up"/>	<input type="button" value="Down"/> <input type="button" value="Delete"/>

Figure 58. Community delivery exchange maintenance page

 A red X indicates a delivery exchange is disabled. If an X is not visible, the exchange is active.

Click **Test** to check whether an exchange point is working as configured. Figure 59 shows the test result for a community exchange for receiving messages from partners.

### Delivery exchange test results

Community: *Worldwide Trading*

Type: EDIINT AS2  
 Transport: Delivery & pickup: Embedded HTTP  
 Location: http://10.20.4.34:4080/exchange/ZZworldwide  
 Name: None  
 Test results: Not applicable

[Close this window](#)

**Figure 59. Test result for a delivery exchange**

For integration pickup exchanges, the user interface displays a maintenance page (Figure 60) with two tabs, one for enabled exchanges and the other for disabled exchanges. This is especially helpful if you have many integration pickup exchanges, but only some are enabled at one time.

### Pick an integration pickup exchange

The following protocols are used to pick up messages from back-end systems. All communities check all of these transports for outbound messages. But a community only picks up the messages that pertain to it and its partners.

<a href="#">Enabled exchanges</a>		<a href="#">Disabled exchanges</a>		
<input type="checkbox"/>	Type	Location	Name	
<input type="checkbox"/>	Other (Plain text) from File system	C:\data\ediout		<a href="#">Test...</a> <a href="#">Delete</a>
<input type="checkbox"/>	Other (Plain text) from File system	C:\data\xmlout		<a href="#">Test...</a> <a href="#">Delete</a>
<input type="checkbox"/>	Other (Plain text) from File system	C:\data\binaryout		<a href="#">Test...</a> <a href="#">Delete</a>

**Figure 60. 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 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

The staged HTTP transport provides a secure way for communities to receive messages from partners without having to change firewall configurations.

This transport is available with the AS2, ebXML, RosettaNet 1.1, RosettaNet 2.0 and secure file 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.

See [Transport guidelines](#) on page 637 for security guidelines for staged HTTP.

## Concepts

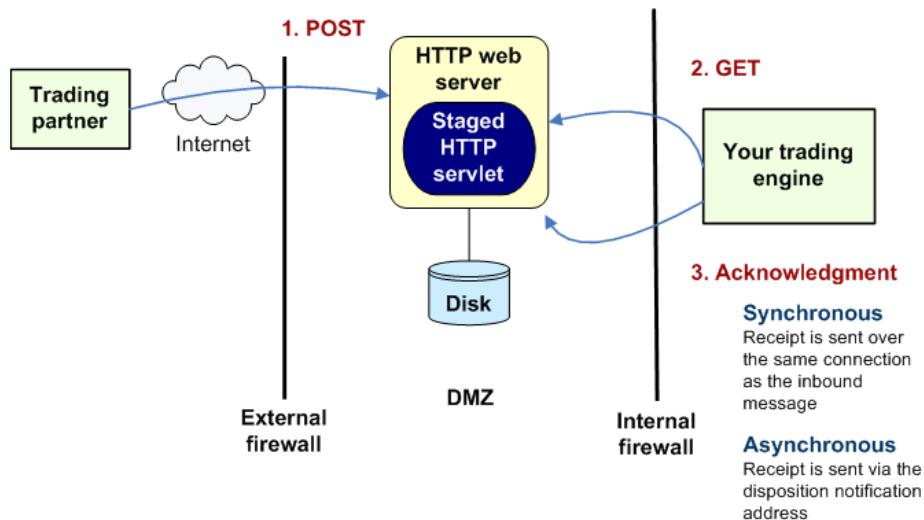
- ◆ [Overview of staged HTTP](#)
- ◆ [Staged HTTP configuration outline](#) on page 278
- ◆ [Staged HTTP files to deploy](#) on page 279
- ◆ [Log on to servlet UI](#) on page 283
- ◆ [Managing mailboxes](#) on page 284
- ◆ [Staged HTTP UI fields](#) on page 288
- ◆ [Message protocols for staged HTTP](#) on page 289
- ◆ [File forwarding to bypass polling](#) on page 291
- ◆ [High availability staged HTTP](#) on page 292

## Procedure

- ◆ [Deploy the web servlet](#) on page 279

## 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 61. Using staged HTTP to receive messages from a partner**

Figure 61 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 (for example, 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 291.)
- 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 based on message protocol configuration.

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 279 and [Deploy the web servlet](#) on page 279.

- 2** Log on to the HTTP servlet user interface. See [Log on to servlet UI on page 283](#).
- 3** Add a mailbox for your partner. See [Add a mailbox on page 284](#).
- 4** For your community profile in the trading engine user interface, add a delivery exchange for receiving messages from the partner. See [Staged HTTP UI fields on page 288](#).

## 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. Archive files in TAR and WAR formats are provided. The file to use depends on the Java Runtime Environment (JRE) version your web server supports and whether you want automatic or manual deployment. A TAR file must be unpackaged and the files manually copied to the web server. A WAR file deploys automatically once copied to the web server.

Use the TAR or WAR file for the JRE your web server supports. For example, if JRE 1.5 is supported, use the TAR or WAR file in the 1.5 subdirectory.

## 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:

- ◆ Apache Tomcat 5.5
- ◆ WebLogic 8.1.4 or later
- ◆ WebLogic 9.0

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	<b>Windows:</b> C:\opt\cyclone\data\stagedhttp  <b>Unix:</b> /opt/cyclone/data/stagedhttp	conf\ configurationrules.xml
Message temporary	<b>Windows:</b> C:\opt\cyclone\tmp\mailbox-data  <b>Unix:</b> /opt/cyclone/tmp/mailbox-data	conf\ webmailboxconfig.xml
Logging	<b>Windows:</b> C:\var\log\cyclone\webmailbox  <b>Unix:</b> /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.

**Note:** If you use WebLogic 9.0, add a subdirectory named logs in the same directory as the WebLogic start-up script. For example, if running the example server that comes with WebLogic, the path for the logs directory would be:

bea/weblogic90/samples/domains/wl\_server/logs

## 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 If you have a Tomcat or Jetty server, have your web server administrator add the following roles in the web server application. The rules for the roles are defined in the servlet's configurationrules.xml file.

**mailboxadmin\_role**

This role can fully configure mailboxes and can change global configuration settings.

**communityadmin\_role**

This role can add, change and delete mailboxes. Mailboxes are created using template rules that the servlet controls. Little deviation from the template is allowed.

**ecengine\_role**

This is the role 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.

After adding the roles, have your web server administrator add the following users in the web server application. The administrator must assign each user to the similarly named role.

mailboxadmin  
communityadmin  
ecengine

- 3** If you have a WebLogic server, 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 community to get documents. This role accesses mailboxes to get messages and then delete the retrieved messages from the staged HTTP servlet.

Partners should not use the ecengine user. Partners can use an anonymous connection. Or, you can add a user to the web server and have one or more partners connect using it. See [Partner connection to staged HTTP](#) on page 286.

- 4 Deploy the stated HTTP servlet on the web server application. The procedure for doing this depends on the web server. Example steps are provided for deploying on Apache Tomcat and WebLogic servers. See the user documentation for your web server for specifics about deploying servlets.

## ***Deploy on Apache Tomcat***

The following are steps for deploying the staged HTTP servlet on an Apache Tomcat web server. These steps are provided as guidelines. See the web server user documentation for specifics about deploying servlets.

### **Steps**

- 1 Copy the WAR file to the Tomcat **webapps** directory. Or, create a subdirectory of **webapps** named **stagedhttp** and copy the extracted TAR files to **webapps\stagedhttp**. See [Staged HTTP files to deploy](#) on page 279 for which archive file to use.

If using a TAR file, extract the file using the tar command on UNIX or using an application such as WinZip on Windows.

- 2 Restart the web server.
- 3 Log on to the staged HTTP servlet user interface. See [Log on to servlet UI](#) on page 283.

## ***Deploy on WebLogic***

The following are steps for deploying the staged HTTP servlet on a WebLogic web server. These steps are provided as guidelines. See the web server user documentation for specifics about deploying servlets.

### **Steps**

- 1 If it does not already exist, create the following home directory:

**UNIX:**

/opt/cyclone

**Windows:**

c:\opt\cyclone

- 2 Create a subdirectory of the home directory named **stagedhttp**.

- 3** Extract the contents of the TAR file and copy the files to the stagedhttp directory. See [Staged HTTP files to deploy](#) on page 279 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.

- 4** Restart the web server.
- 5** 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.
- 6** Log on to the staged HTTP servlet user interface. See [Log on to servlet UI](#) on page 283.

## 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:** Use **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. If the port is 80, you do not have to include it in the URL.

Type the user name and password of the mailbox administrator when prompted.

The browser displays the staged HTTP servlet page in Figure 62.



**Figure 62. 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.

The screenshot shows the 'Mailbox Maintenance' page of the cyclone commerce application. At the top, there's a red header bar. Below it, the cyclone commerce logo and the company name 'The Collaborative Commerce Company'. The main area has a white background with several input fields and a table.

**Configuration Fields:**

- URI: [redacted]
- Company Name: [redacted]
- File System Directory: /opt/cyclone/data/stagedhttp/
- URL: http://windows10/webmailbox/

**Privilege Table:**

	Name	Type	# Instances	Allow Read	Allow Write
<input type="checkbox"/>	anonymous	USER	1000000	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	ecengine	USER	1000000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

At the bottom of the page are buttons: Add Privilege, Delete Privilege, Save Mailbox, and Close.

**Figure 63. 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.

## **Partner connection to staged HTTP**

Partners can use an anonymous connection to write messages to the staged HTTP server. Or, you can add a user to the web server and have one or more partners connect using it. Partners should not use the ecengine user to connect.

The partner's connection to the web server is not authenticated when anonymous is used. This means anyone connecting anonymously could post data to the web server. Whether you want partners to connect anonymously or want to add a user on the web server for use by partners depends on your organization's security policies.

Allowing an anonymous user to write to a mailbox requires changing the stagedhttp/WEB-INF/web.xml file where the staged HTTP servlet is deployed on the web server (Figure 64).

---

```
<web-app>
  ...
  <security-constraint>
    ...
    <web-resource-collection>
      <web-resource-name>All Pages</web-resource-name>
        <url-pattern>/*</url-pattern>
        <http-method>DELETE</http-method>
        <http-method>GET</http-method>
        <http-method>PUT</http-method>
        <http-method>POST</http-method>
    </web-resource-collection>
    ...
  </security-constraint>
  ...
</web-app>
```

---

**Figure 64. web.xml file before commenting out POST method**

Comment out the specification of the POST method (Figure 65). Restart the servlet after making this change.

---

```

<web-app>
...
<security-constraint>
...
<web-resource-collection>
    <web-resource-name>All Pages</web-resource-name>
        <url-pattern>/*</url-pattern>
        <http-method>DELETE</http-method>
        <http-method>GET</http-method>
        <http-method>PUT</http-method>
        <!-- <http-method>POST</http-method> -->
    </web-resource-collection>
...
</security-constraint>
...
</web-app>

```

---

**Figure 65. web.xml file after commenting out POST method**

If you comment out the POST method, all POST requests appear to the staged HTTP servlet to come from the anonymous user, even when a valid user name and password are specified. If this is not desired, a user name and password must always be used to send to the staged HTTP servlet. It is not recommended to use the standard ecengine user for this purpose, as that user also has permissions to read from the staged HTTP servlet. A new user can be used for the purpose of sending to the staged HTTP servlet, as long as the user and its role are configured properly in the web server. If a new role is configured, edit stagedhttp/WEB-INF/web.xml and add the role to the auth-constraint element in the security-constraint (Figure 66).

---

```

<web-app>
...
<security-constraint>
...
<auth-constraint>
    <role-name>mailboxadmin_role</role-name>
    <role-name>communityadmin_role</role-name>
    <role-name>ecengine_role</role-name>
    <role-name>anonymous</role-name>
    <!-- Add role-name element for new role here. -->
</auth-constraint>
...
</security-constraint>
...
</web-app>

```

---

**Figure 66. auth-constraint section of web.xml file**

## **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.

**Note:** If you use Interchange or Activator version 5 or later do not select the check box for “Directories accessible by the trading engine?” Version 5 or later does not support this. This applies only to trading engines that poll files from the staged HTTP servlet.

## Staged HTTP UI fields

The following fields are used in the delivery exchange wizard in the trading engine user interface for configuring this transport. See [Message protocols for staged HTTP](#) on page 289 for information about using staged HTTP with the AS2, ebXML, RosettaNet 1.1, RosettaNet 2.0 and secure file message protocols.

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.



Figure 67. Add staged HTTP 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 not encrypted.

***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 to connect to the server.

# Message protocols for staged HTTP

The staged HTTP transport is supported for the AS2, ebXML, RosettaNet 1.1, RosettaNet 2.0 and secure file message protocols. The following topics provide notes about using each protocol with staged HTTP. Extra configuration steps are required only with the secure file message protocol.

## AS2

Using staged HTTP to trade AS2 messages does not require any special configuration of the staged HTTP servlet. The servlet, in conjunction with the trading engine, knows how to handle asynchronous and synchronous receipts correctly.

## ebXML

Using staged HTTP to trade ebXML messages does not require any special configuration of the staged HTTP servlet. The servlet, in conjunction with the trading engine, knows how to handle asynchronous and synchronous acknowledgments correctly.

## RosettaNet 1.1 and 2.0

Using staged HTTP to trade RosettaNet 1.1 and 2.0 messages does not require any special configuration of the staged HTTP servlet. However, the trading engine does not support synchronous RosettaNet 2.0 responses of any type (single or dual action). Consequently, synchronous RosettaNet 2.0 responses over staged HTTP are not supported.

## Secure file

Supporting the secure file message protocol requires special configuration of the staged HTTP servlet.

The configuration requires editing two files in the staged HTTP servlet's conf directory on the application server: configurationrules.xml and webmailboxconfig.xml. After editing the files, restart the servlet.

Support for the secure file protocol by the staged HTTP servlet is on a mailbox-by-mailbox basis. To have all mailboxes added from this point forward support the secure file protocol, open the configurationrules.xml file and add or change the DefaultPostMode element to the Mailbox element inside the MailboxTemplate element (Figure 68).

---

```
<Rules>
  ...
  <MailboxTemplate id="default">
    <Mailbox>
      ...
      <DefaultPostMode>async</DefaultPostMode>
      ...
    </Mailbox>
  </MailboxTemplate>
  ...
</Rules>
```

---

**Figure 68. configurationrules.xml file**

To add support for the secure file protocol to specific existing mailboxes, open the webmailboxconfig.xml file and add or change the DefaultPostMode element in each desired Mailbox element in the Mailboxes element (Figure 69).

---

```
<WebMailboxConfiguration>
  ...
  <Mailboxes>
    ...
    <Mailbox ...>
      ...
      <DefaultPostMode>async</DefaultPostMode>
      ...
    </Mailbox>
    ...
  </Mailboxes>
  ...
</WebMailboxConfiguration>
```

---

**Figure 69. webmailboxconfig.xml file**

# 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="/SOMEEMAILBOX"`

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.

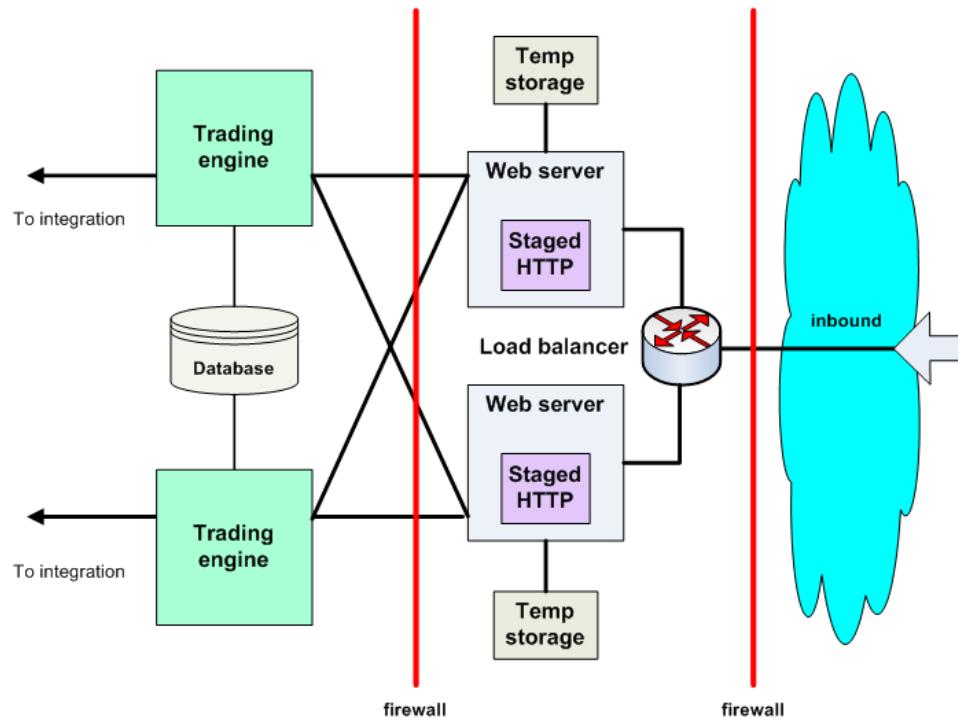
**<ResponseTimeout>600000</ResponseTimeout>**

The time in milliseconds the sender waits for a response before dropping the connection.

## High availability staged HTTP

Figure 70 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 70. 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 296

## Procedure

- ◆ [Change embedded transports](#) on page 375

## Transport pages and fields

- ◆ [SMTP maintenance](#) on page 296
- ◆ [HTTP maintenance](#) on page 305
- ◆ [Staged HTTP maintenance](#) on page 313
- ◆ [External FTP maintenance](#) on page 316
- ◆ [Embedded FTP maintenance](#) on page 325
- ◆ [SFTP maintenance](#) on page 331
- ◆ [File system maintenance](#) on page 338
- ◆ [JMS maintenance](#) on page 342
- ◆ [MQSeries maintenance](#) on page 349
- ◆ [Web services integration maintenance](#) on page 354
- ◆ [Web services trading maintenance](#) on page 358

## Common tabs

- ◆ [Proxy tab](#) on page 362
- ◆ [From address and To address tabs](#) on page 362
- ◆ [Message attributes tab](#) on page 364
- ◆ [EDI Splitter tab](#) on page 370
- ◆ [Inline processing tab](#) on page 371
- ◆ [Schedule tab](#) on page 371
- ◆ [Delivery criteria tab](#) on page 373

# 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 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 375.

### E-mail address

The e-mail address that the remote partner uses to send messages to your local community.

## Advanced tab (Embedded SMTP settings)

### Include attachments only

This check box is applicable when you have partners who send messages using a mail client such as Microsoft Outlook. The control lets you eliminate extraneous message fragments. This control is available under the generic e-mail message protocol, but not AS1.

When a partner uses a mail client application to send a trading document as an attachment to an e-mail message, the trading engine actually receives two or more documents. These can include the MIME header, the text of the e-mail message and the document attachment. The trading engine tracks and processes the incidental MIME body parts just as it does any document. Although such processing does no harm, it can cause confusion.

Selecting this check box causes the incidental MIME body parts to be ignored while preserving the important document attachments.

### 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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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.

### **Restrict maximum file size for this transport**

Optionally lets you specify the maximum size of files a transport can handle.

If the trading engine receives a file larger than the maximum, the file is rejected and a message is written to the events log. If received via HTTP, a 413 response also is sent and the connection is closed. A 413 message is Request Entity Too Large.

The maximum size must be expressed in bytes. Do not use commas. For instance, a kilobyte is 1024 bytes, a megabyte is 1048576 bytes, a gigabyte is 1073741824 bytes.

The smallest maximum allowed is 1000 bytes. On the opposite extreme, you can enter the largest number the field can accommodate.

This control is available only for transports used for picking up messages from integration or receiving messages from partners.

## **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 23.

## 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 concurrent connections

The maximum number of simultaneous connections allowed from partners. If the trading engine has multiple processing nodes, each instance of the server running on different computers can host this many connections.

### 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)**

Time in seconds the trading engine waits for a connection to the delivery exchange before the attempt times out. Although the default value is 30 seconds, this may be longer than the interval allowed by your operating system (OS). For example, Windows XP by default allows a maximum timeout of 20 seconds. The actual connect timeout interval is the lesser of the OS timeout and the value set in the trading engine.

**Read timeout (seconds)**

Time in seconds 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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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)**

Time in seconds the trading engine waits for a connection to the delivery exchange before the attempt times out. Although the default value is 30 seconds, this may be longer than the interval allowed by your operating system (OS). For example, Windows XP by default allows a maximum timeout of 20 seconds. The actual connect timeout interval is the lesser of the OS timeout and the value set in the trading engine.

### **Read timeout (seconds)**

Time in seconds the trading engine waits to read data from the delivery exchange before terminating the connection.

If you handle files larger than 200 megabytes, it is recommended to set this to a minimum of 150 seconds (2.5 minutes), but preferably to 300 seconds (5 minutes).

**Include attachments only**

This check box is applicable when you have partners who send messages using a mail client such as Microsoft Outlook. The control lets you eliminate extraneous message fragments. This control is available under the generic e-mail message protocol, but not AS1.

When a partner uses a mail client application to send a trading document as an attachment to an e-mail message, the trading engine actually receives two or more documents. These can include the MIME header, the text of the e-mail message and the document attachment. The trading engine tracks and processes the incidental MIME body parts just as it does any document. Although such processing does no harm, it can cause confusion.

Selecting this check box causes the incidental MIME body parts to be ignored while preserving the important document attachments.

**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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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.

**Restrict maximum file size for this transport**

Optionally lets you specify the maximum size of files a transport can handle.

If the trading engine receives a file larger than the maximum, the file is rejected and a message is written to the events log. If received via HTTP, a 413 response also is sent and the connection is closed. A 413 message is Request Entity Too Large.

The maximum size must be expressed in bytes. Do not use commas. For instance, a kilobyte is 1024 bytes, a megabyte is 1048576 bytes, a gigabyte is 1073741824 bytes.

The smallest maximum allowed is 1000 bytes. On the opposite extreme, you can enter the largest number the field can accommodate.

This control is available only for transports used for picking up messages from integration or receiving messages from partners.

#### **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.

#### **Specify preferred nodes**

If there are one or more nodes for the trading engine, you can select one or more as the preferred nodes for consuming messages. If the preferred nodes are running, these are used to process messages. If the preferred nodes are stopped, work is distributed among the remaining running available nodes. Selecting preferred nodes lets you manage work distribution among nodes.

This option is available for integration pickup and trading delivery exchanges that poll for messages.

## **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 388.

When the trading engine sends a message via HTTP, it is the HTTP client. The endpoint the trading engine is connecting to is the HTTP server.

HTTP error codes are reported in Activator log files. When you see an HTTP code in a log file, the trading engine merely is echoing the code returned by an HTTP server, firewall, proxy server or load balancer.

The response code for a successfully sent message is 200 (meaning “OK”). When a 503 code is returned, this may be due to action by the trading engine system throttle. A 503 code indicates service temporarily is unavailable. The throttle attempts to monitor the message load by looking at available memory and the depth of internal queues. When a loaded condition is detected, consumption of new work is halted. This means two things: exchange points stop polling and embedded servers return a 503 code.

For a complete list of HTTP codes, go to <http://www.ietf.org/rfc/rfc2616.txt> and see section 10 of the document.

## 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. When you export your community profile as a partner profile, the URL becomes part of your exported partner profile. 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 path 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. When you export your community profile as a partner profile, the URL becomes part of your exported partner profile. 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 path 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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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.

### Restrict maximum file size for this transport

Optionally lets you specify the maximum size of files a transport can handle.

If the trading engine receives a file larger than the maximum, the file is rejected and a message is written to the events log. If received via HTTP, a 413 response also is sent and the connection is closed. A 413 message is Request Entity Too Large.

The maximum size must be expressed in bytes. Do not use commas. For instance, a kilobyte is 1024 bytes, a megabyte is 1048576 bytes, a gigabyte is 1073741824 bytes.

The smallest maximum allowed is 1000 bytes. On the opposite extreme, you can enter the largest number the field can accommodate.

This control is available only for transports used for picking up messages from integration or receiving messages from partners.

## 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 not encrypted.

### *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 to connect to the server.

## Advanced tab (partner)

### Maximum concurrent connections

The maximum number of simultaneous connections allowed from partners. If the trading engine has multiple processing nodes, each instance of the server running on different computers can host this many connections.

## 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)**

Time in seconds the trading engine waits for a connection to the delivery exchange before the attempt times out. Although the default value is 30 seconds, this may be longer than the interval allowed by your operating system (OS). For example, Windows XP by default allows a maximum timeout of 20 seconds. The actual connect timeout interval is the lesser of the OS timeout and the value set in the trading engine.

### **Read timeout (seconds)**

Time in seconds 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.

### **Enable HTTP chunking**

If you are sending files larger than 2 gigabytes, select this to turn on chunking. A chunked message is a large message broken into smaller pieces for sending to a partner over the Internet or to back-end integration.

Although primarily for handling large messages, chunking can be enabled for small messages, too. However, if your partners use a trading engine other than Interchange or Activator or use an external or staged HTTP server, they may be unable to accept messages larger than 2 gigabytes, even if the messages are chunked.

Also, in rare cases a partner's HTTP server may be unable to handle chunked messages, regardless of message size. You should perform tests to determine whether a partner's server can handle chunked messages. If not, the partner must use Interchange or Activator with the embedded server to receive large chunked messages successfully.

If you enable chunking because of large messages, you also probably need to request that receipts be sent over an asynchronous connection. See the chapter on collaboration settings for details.

### Attempt restarts

Select this to turn on checkpoint-restart. A checkpoint is information saved to disk that is a recovery point in the event of a transport failure. The restart program uses the last saved checkpoint and starts again, ensuring no loss of data.

The checkpoint files are saved on the server under the [install directory]/common/data/http/restartable, which is normally common to all nodes in the cluster. Thus, if a transfer is interrupted and the load balancer directs the restart request to a different node, the restart file will be accessible to the new node even though it did not process the original request.

To reduce unnecessary overhead when processing small files, checkpoint files are only created for messages that are at least 100 KB in size. Also, if a restart is attempted for a message whose checkpoint file on the server is more than four hours old, the checkpoint file will be discarded and the entire message will be retransmitted.

The restart logic is used only during transport retries, such as might occur when a transfer is interrupted due to network problems. If you resubmit a message in Message Tracker, no attempt is made to perform a checkpoint-restart.

This feature only works if your partner uses Interchange or Activator and its embedded HTTP server. Do not select this option if a partner uses an external or staged HTTP server or uses a trading engine other than Interchange or Activator.

### Enable use of 102-processing

This check box is available to ensure the connection between the client and server does not become idle and fail while message processing is in progress. For example, this makes sure the connection remains active when the client is sending a multi-gigabyte message. Or, to prevent a firewall from disconnecting an idle connection before the server receives the entire message and returns a 200 OK response. Most often this setting is useful when the client requests a synchronous receipt, but also could be recommended in some cases for an asynchronous receipt.

Selecting this check box directs the trading engine to add the following to the header of an outbound message: Expect: 102-processing. This is an HTTP response code that indicates processing is in progress. If the receiving server supports 102 responses, the header triggers the server to send 102 responses to the client repeatedly until the server has completely processed the inbound message.

Before selecting this option, make sure the server supports 102 responses. If you turn on 102 processing and the server does not support it, the server will return a 417 message (the server could not meet the expectation given in the Expect header) and the connection may fail. If the receiving partner uses the embedded HTTP server in Interchange or Activator 5.5.1 or later, 102 responses are supported. This also is supported if your partner uses Jetty 6 or later.

#### **Receiver is Synchrony Gateway**

This field appears only when an HTTP or HTTPS transport under the secure file message protocol has been added to a partner profile.

Selecting this check box enables the trading engine to send messages successfully to Synchrony Gateway via secure file HTTP.

Upon receiving a secure file HTTP message from Interchange or Activator, Synchrony Gateway expects to find the payload's file name in the HTTP POST request URL. Normally, the trading engine does not include this. But when the check box is selected, the trading engine appends **?filename=name** to the URL, where **name** is the production file name.

#### **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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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 to connect to the server.

### HTTPS settings tab

#### URL

The URL for connecting to the server.

#### 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 to connect to the server.

## Advanced tab

### Maximum concurrent connections

The maximum number of simultaneous connections allowed from partners. If the trading engine has multiple processing nodes, each instance of the server running on different computers can host this many connections.

### 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)**

Time in seconds the trading engine waits for a connection to the delivery exchange before the attempt times out. Although the default value is 30 seconds, this may be longer than the interval allowed by your operating system (OS). For example, Windows XP by default allows a maximum timeout of 20 seconds. The actual connect timeout interval is the lesser of the OS timeout and the value set in the trading engine.

#### **Read timeout (seconds)**

Time in seconds 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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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.

**Restrict maximum file size for this transport**

Optionally lets you specify the maximum size of files a transport can handle.

If the trading engine receives a file larger than the maximum, the file is rejected and a message is written to the events log. If received via HTTP, a 413 response also is sent and the connection is closed. A 413 message is Request Entity Too Large.

The maximum size must be expressed in bytes. Do not use commas. For instance, a kilobyte is 1024 bytes, a megabyte is 1048576 bytes, a gigabyte is 1073741824 bytes.

The smallest maximum allowed is 1000 bytes. On the opposite extreme, you can enter the largest number the field can accommodate.

This control is available only for transports used for picking up messages from integration or receiving messages from partners.

**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.

**Specify preferred nodes**

If there are one or more nodes for the trading engine, you can select one or more as the preferred nodes for consuming messages. If the preferred nodes are running, these are used to process messages. If the preferred nodes are stopped, work is distributed among the remaining running available nodes. Selecting preferred nodes lets you manage work distribution among nodes.

This option is available for integration pickup and trading delivery exchanges that poll for messages.

# External FTP maintenance

The following topics document the fields on the maintenance pages for the external FTP transport. This is for FTP servers outside of the trading engine.

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 388.

## 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 (embedded FTP default is 4021).

### 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 not encrypted.

#### ***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.

### **Directories tab**

#### **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.

If connecting to a partner's Activator embedded FTP server, use **mailbox** if picking up. Leave the field blank if delivering.

#### **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.

If connecting to a partner's Activator embedded FTP server, clear the check box.

#### ***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 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.

#### **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.

## Filenames tab

***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.

For most transports, the appended number is consecutively numbered. For FTP and SFTP, however, the appended number is hexadecimal and looks this: filename\_c4.

### 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

### Make output filenames safe for all FTP servers

Select this check box to have the trading engine add a letter prefix to file names that begin with a numeral. This is for servers that cannot process files with names that begin with a numeral.

## Advanced tab

### Maximum concurrent connections

The maximum number of simultaneous connections allowed from partners. If the trading engine has multiple processing nodes, each instance of the server running on different computers can host this many connections.

### 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)

Time in seconds the trading engine waits for a connection to the delivery exchange before the attempt times out. Although the default value is 30 seconds, this may be longer than the interval allowed by your operating system (OS). For example, Windows XP by default allows a maximum timeout of 20 seconds. The actual connect timeout interval is the lesser of the OS timeout and the value set in the trading engine.

### Read timeout (seconds)

Time in seconds 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 key

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"/>
```

#### **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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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.

#### **Restrict maximum file size for this transport**

Optionally lets you specify the maximum size of files a transport can handle.

If the trading engine receives a file larger than the maximum, the file is rejected and a message is written to the events log. If received via HTTP, a 413 response also is sent and the connection is closed. A 413 message is Request Entity Too Large.

The maximum size must be expressed in bytes. Do not use commas. For instance, a kilobyte is 1024 bytes, a megabyte is 1048576 bytes, a gigabyte is 1073741824 bytes.

The smallest maximum allowed is 1000 bytes. On the opposite extreme, you can enter the largest number the field can accommodate.

This control is available only for transports used for picking up messages from integration or receiving messages from partners.

**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.

**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.

**Specify preferred nodes**

If there are one or more nodes for the trading engine, you can select one or more as the preferred nodes for consuming messages. If the preferred nodes are running, these are used to process messages. If the preferred nodes are stopped, work is distributed among the remaining running available nodes. Selecting preferred nodes lets you manage work distribution among nodes.

This option is available for integration pickup and trading delivery exchanges that poll for messages.

# Embedded FTP maintenance

The following are the fields on the maintenance pages for the embedded FTP transport.

In addition, the following topics provide information for managing embedded FTP transports.

- ◆ [Managing users of embedded FTP on page 329](#)
- ◆ [Programmatically submit via embedded FTP on page 330](#)

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 FTP settings tab

### Name

A link is provided to view the settings for the embedded server. You also can change servers.

### Host

The name used by the trading engine for the computer running the embedded server. You cannot change this field.

### Host used by partners

This is the fully qualified domain name or IP address your partners use to send messages to this delivery exchange. When you export your community profile as a partner profile, the host information becomes part of your exported partner profile.

You can change this field by clicking **View settings for this embedded server** and changing the External host or IP address field. If your network uses a load balancer or firewall, contact your network administrator for the correct value. Any change to this field affects all delivery exchanges that reference the server.

### 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.

For most transports, the appended number is consecutively numbered. For FTP and SFTP, however, the appended number is hexadecimal and looks this: filename\_c4.

**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:

dabeeed45\_4cb.edi  
z47e4120\_4ce

**FTP users tab (community)**

This tab enables you to modify or delete the following.

**Owning party**

The community or partner associated with the user account.

If this party is deleted, the FTP account and all references to it associated with exchanges are deleted.

**FTP user**

The name of the user.

**Directory path**

The FTP directory associated with the user. A specific combination of user and directory can be associated with only one exchange.

**Specify what to do when partners upload messages to subdirectories of the directories listed above**

The **Allow** option allows the user to write files to any subdirectory under the root path.

The **Do not allow** option allows writing files to a subdirectory under the root path only when a message attribute is set up for each subdirectory level. See [Message attributes tab](#) on page 364.

## FTP users tab (integration)

This tab is also the same for integration pickup.

This tab enables you to modify or delete the following.

**FTP user**

The name of the user.

**Directory path**

The FTP directory associated with the user. A specific combination of user and directory can be associated with only one exchange.

## Advanced tab

**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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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.

### Restrict maximum file size for this transport

Optionally lets you specify the maximum size of files a transport can handle.

If the trading engine receives a file larger than the maximum, the file is rejected and a message is written to the events log. If received via HTTP, a 413 response also is sent and the connection is closed. A 413 message is Request Entity Too Large.

The maximum size must be expressed in bytes. Do not use commas. For instance, a kilobyte is 1024 bytes, a megabyte is 1048576 bytes, a gigabyte is 1073741824 bytes.

The smallest maximum allowed is 1000 bytes. On the opposite extreme, you can enter the largest number the field can accommodate.

This control is available only for transports used for picking up messages from integration or receiving messages from partners.

### 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.

## ***Managing users of embedded FTP***

There are various places in the user interface where you can view user accounts associated with embedded FTP servers. The separate areas reflect the three distinct types of accounts:

- ◆ [Community FTP accounts](#)
- ◆ [Partner FTP accounts](#)
- ◆ [Integration FTP accounts](#)

There are different types of accounts because a single embedded FTP server can be used for integration or trading but not both. Separate embedded FTP servers are required. For more information, see [Embedded FTP transport](#) on page 230 and [FTP user accounts](#) on page 232.

On the FTP user pages you can change the password for a user and enable or disable a user. You also can click a link to the exchange point associated with a user. An FTP user account can be deleted only if it is not associated with a delivery exchange.

Attempting to log on to a disabled account results in a 421 account disabled message.

## View community FTP accounts

To view community FTP accounts, open the summary page for a community and click **FTP users** in the navigation graphic at the top of the page. The FTP users icon displays only if an embedded FTP has been added for a community.

## View integration FTP accounts

To view integration FTP accounts, open the summary page for a community and click **Integration delivery** or **Integration pickup** in the navigation graphic at the top of the page. Depending on which icon you click, a page displays listing integration delivery exchanges or integration pickup exchanges. Click **Manage integration FTP users** in the task list at the bottom of the page. The link displays only if an embedded FTP integration exchange has been added.

## View partner FTP accounts

To view partner FTP accounts, open the summary page for a partner and click **FTP users** in the navigation graphic at the top of the page. The FTP users icon displays only if FTP users owned by a partner have been defined for a community or partner embedded FTP exchange.

# **Programmatically submit via embedded FTP**

The following is an example of a Java method to programmatically submit a message to the trading engine through an embedded FTP exchange.

```
/**  
 * Java method to submit a message to an Interchange embedded FTP  
server. Typically a back-end system might do  
 * this to submit a message to an integration pickup exchange, but a  
partner could also use this to submit a  
 * message to a community exchange on the trading side. In this example  
the message is dropped off to a  
 * subdirectory to identify the sender and receiver. This is necessary  
for trading files that contain no  
 * sender/receiver information. On the trading side it wouldn't be  
necessary if the protocol (e.g. AS3)  
 * identifies the sender and receiver.  
 */  
private void submitFtp(byte[] message) throws IOException  
{  
    // User/pw are both "integration". Assumes 2 levels of msg attr  
mapping under "out": lev1=sender, lev2=rcvr
```

```
URL url = new URL("ftp://integration:integration@localhost:5021/  
out/ed/anne");  
FtpURLConnection con = new FtpURLConnection(url);  
OutputStream os = con.getOutputStream();  
os.write(message);  
os.close();  
}
```

# 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.

### 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

Select this check box to display options for designating a new RSA or DSA public key for the SFTP server. The trading engine uses the key to authenticate the server. If the server is modified to use a new public-private key pair, the public key must be updated.

#### *Retrieve public key from server*

Click **Get Key** to have the trading engine retrieve the public key for the SFTP server. The server name and port number entered on this page must be correct for this option to work.

**Server public key file**

Type the path to the file containing the public key for the SFTP server or click **Browse** to locate the file. You may have to ask the server administrator for the file path.

**Authentication method**

There are three methods.

**Password** requires entering the user name and password for connecting to the server. The user name and password are sent over an encrypted connection to authenticate the user to the server. Although this option offers ease of administration, the password is vulnerable because it is sent every time a connection is made. The password could be compromised if the server is ever compromised.

**Hostbased** requires entering the user name of the server, the path to the private key and the password for the private key, if any. Authentication relies on the private key on the client side and the public key on the server side. With this option, one public-private key pair is shared for all users that connect from a single client. Care is advised to make sure a regular user does not have access to the private key.

**Public key** requires entering the user name of the server and the path to the private key. A password for the private key is optional but recommended. Authentication relies on the private key on the client side and the public key on the server side. Public key authentication requires administrators to create public-private key pairs for each user and then place the public key on the host in each user's home directory.

**Directories tab****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 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.

## Filenames tab

### **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.

For most transports, the appended number is consecutively numbered. For FTP and SFTP, however, the appended number is hexadecimal and looks this: filename\_c4.

### **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:

dabeeed45\_4cb.edi  
z47e4120\_4ce

#### **Make output filenames safe for all FTP servers**

Select this check box to have the trading engine add a letter prefix to file names that begin with a numeral. This is for servers that cannot process files with names that begin with a numeral.

### **Advanced tab**

#### **Maximum concurrent connections**

The maximum number of simultaneous connections allowed from partners. If the trading engine has multiple processing nodes, each instance of the server running on different computers can host this many connections.

#### **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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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.

**Restrict maximum file size for this transport**

Optionally lets you specify the maximum size of files a transport can handle.

If the trading engine receives a file larger than the maximum, the file is rejected and a message is written to the events log. If received via HTTP, a 413 response also is sent and the connection is closed. A 413 message is Request Entity Too Large.

The maximum size must be expressed in bytes. Do not use commas. For instance, a kilobyte is 1024 bytes, a megabyte is 1048576 bytes, a gigabyte is 1073741824 bytes.

The smallest maximum allowed is 1000 bytes. On the opposite extreme, you can enter the largest number the field can accommodate.

This control is available only for transports used for picking up messages from integration or receiving messages from partners.

**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.

**Specify preferred nodes**

If there are one or more nodes for the trading engine, you can select one or more as the preferred nodes for consuming messages. If the preferred nodes are running, these are used to process messages. If the preferred nodes are stopped, work is distributed among the remaining running available nodes. Selecting preferred nodes lets you manage work distribution among nodes.

This option is available for integration pickup and trading delivery exchanges that poll for messages.

# 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. When adding a file system transport, the delivery exchange wizard suggests a 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.

For most transports, the appended number is consecutively numbered. For FTP and SFTP, however, the appended number is hexadecimal and looks this: filename\_c4.

**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:

dabeeed45\_4cb.edi  
z47e4120\_4ce

**Advanced tab****Maximum concurrent connections**

The maximum number of simultaneous connections allowed from partners. If the trading engine has multiple processing nodes, each instance of the server running on different computers can host this many connections.

## 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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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.

#### **Restrict maximum file size for this transport**

Optionally lets you specify the maximum size of files a transport can handle.

If the trading engine receives a file larger than the maximum, the file is rejected and a message is written to the events log. If received via HTTP, a 413 response also is sent and the connection is closed. A 413 message is Request Entity Too Large.

The maximum size must be expressed in bytes. Do not use commas. For instance, a kilobyte is 1024 bytes, a megabyte is 1048576 bytes, a gigabyte is 1073741824 bytes.

The smallest maximum allowed is 1000 bytes. On the opposite extreme, you can enter the largest number the field can accommodate.

This control is available only for transports used for picking up messages from integration or receiving messages from partners.

#### **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.

**Specify preferred nodes**

If there are one or more nodes for the trading engine, you can select one or more as the preferred nodes for consuming messages. If the preferred nodes are running, these are used to process messages. If the preferred nodes are stopped, work is distributed among the remaining running available nodes. Selecting preferred nodes lets you manage work distribution among nodes.

This option is available for integration pickup and trading delivery exchanges that poll for messages.

## 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.

If you have set up a Synchrony Integrator integration exchange for performing JMS integration with Synchrony Integrator, review this maintenance topic. The Synchrony Integrator transport is a special JMS exchange for integrating the trading engine and Synchrony Integrator. For more information see [Integrate with Synchrony Integrator](#) on page 661.

### 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@routernname** 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.

**Database name.** The name of the database that contains the 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**.

### Send payload via file system

Select this check box to have payloads sent by a file system. You can specify the size of payloads to send and the path for payload files. The receiver uses the path to retrieve the files.

This option is available when the exchange is for integration delivery and sending to partners.

## Advanced tab (JMS polled)

### Maximum concurrent connections

The maximum number of simultaneous connections allowed from partners. If the trading engine has multiple processing nodes, each instance of the server running on different computers can host this many connections.

### 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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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.

#### **Restrict maximum file size for this transport**

Optionally lets you specify the maximum size of files a transport can handle.

If the trading engine receives a file larger than the maximum, the file is rejected and a message is written to the events log. If received via HTTP, a 413 response also is sent and the connection is closed. A 413 message is Request Entity Too Large.

The maximum size must be expressed in bytes. Do not use commas. For instance, a kilobyte is 1024 bytes, a megabyte is 1048576 bytes, a gigabyte is 1073741824 bytes.

The smallest maximum allowed is 1000 bytes. On the opposite extreme, you can enter the largest number the field can accommodate.

This control is available only for transports used for picking up messages from integration or receiving messages from partners.

#### **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.

### Specify preferred nodes

If there are one or more nodes for the trading engine, you can select one or more as the preferred nodes for consuming messages. If the preferred nodes are running, these are used to process messages. If the preferred nodes are stopped, work is distributed among the remaining running available nodes. Selecting preferred nodes lets you manage work distribution among nodes.

This option is available for integration pickup and trading delivery exchanges that poll for messages.

## 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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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.

### Restrict maximum file size for this transport

Optionally lets you specify the maximum size of files a transport can handle.

If the trading engine receives a file larger than the maximum, the file is rejected and a message is written to the events log. If received via HTTP, a 413 response also is sent and the connection is closed. A 413 message is Request Entity Too Large.

The maximum size must be expressed in bytes. Do not use commas. For instance, a kilobyte is 1024 bytes, a megabyte is 1048576 bytes, a gigabyte is 1073741824 bytes.

The smallest maximum allowed is 1000 bytes. On the opposite extreme, you can enter the largest number the field can accommodate.

This control is available only for transports used for picking up messages from integration or receiving messages from partners.

## 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 connection type

Select **Client connection** to use a channel connection, on the local machine or via the network, to connect to a queue manager.

Select **Server binding** to use an API connection, via shared memory, to a local queue manager.

#### 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.

**Convert data**

Select to convert the characters set of messages received from a queue to the set specified in the Characters set field. Clear the check box to turn off data conversion. This setting does not apply to messages outbound to a queue.

**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.

**Message segmentation**

Select **MQSeries segmentation** to segment messages into chunks that together equal the maximum message length value of the queue as set by the queue administrator.

Select **Application segmentation** to segment messages into chunks equal to the value specified in the Segmentation size field. Each segment must be equal to or less than the maximum message length value of the queue.

Application segmentation can be used when queue manager segmentation does not suffice because messages are too large to be handled by the application in a single buffer. Or, when data conversion must be performed by sender channels and the format is such that the putting application must specify the segment boundaries to make possible the conversion of individual segments.

**This server requires a user name and password**

If selected, type a user name and password to connect to the server.

## Message persistence

Applicable only for outbound messages, specifies whether messages are persisted in the event of system failures or restarts.

**Non-persisted** means messages probably will be lost.

**Persisted** means messages will survive failures or restarts. This overrides the persistence configured for the queue.

**As defined by the queue** means messages are persisted according to the queue configuration.

## Advanced tab

### Maximum concurrent connections

The maximum number of simultaneous connections allowed from partners. If the trading engine has multiple processing nodes, each instance of the server running on different computers can host this many connections.

### 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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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.

**Restrict maximum file size for this transport**

Optionally lets you specify the maximum size of files a transport can handle.

If the trading engine receives a file larger than the maximum, the file is rejected and a message is written to the events log. If received via HTTP, a 413 response also is sent and the connection is closed. A 413 message is Request Entity Too Large.

The maximum size must be expressed in bytes. Do not use commas. For instance, a kilobyte is 1024 bytes, a megabyte is 1048576 bytes, a gigabyte is 1073741824 bytes.

The smallest maximum allowed is 1000 bytes. On the opposite extreme, you can enter the largest number the field can accommodate.

This control is available only for transports used for picking up messages from integration or receiving messages from partners.

**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.

**Specify preferred nodes**

If there are one or more nodes for the trading engine, you can select one or more as the preferred nodes for consuming messages. If the preferred nodes are running, these are used to process messages. If the preferred nodes are stopped, work is distributed among the remaining running available nodes. Selecting preferred nodes lets you manage work distribution among nodes.

This option is available for integration pickup and trading delivery exchanges that poll for messages.

# Web services integration maintenance

The following topics document the fields on the maintenance pages for the web services API integration transport.

**Note:** This topic covers maintenance of the web services integration transport. For maintaining the web services transport for exchanging messages between a community and partners, see [Web services trading maintenance](#) on page 358.

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 tab (web services API client)

### Maximum concurrent connections

The maximum number of simultaneous connections allowed from partners. If the trading engine has multiple processing nodes, each instance of the server running on different computers can host this many connections.

### 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.

### **Enable HTTP chunking**

If you are sending files larger than 2 gigabytes, select this to turn on chunking. A chunked message is a large message broken into smaller pieces for sending to a partner over the Internet or to back-end integration.

Although primarily for handling large messages, chunking can be enabled for small messages, too. However, if your partners use a trading engine other than Interchange or Activator or use an external or staged HTTP server, they may be unable to accept messages larger than 2 gigabytes, even if the messages are chunked.

Also, in rare cases a partner's HTTP server may be unable to handle chunked messages, regardless of message size. You should perform tests to determine whether a partner's server can handle chunked messages. If not, the partner must use Interchange or Activator with the embedded server to receive large chunked messages successfully.

If you enable chunking because of large messages, you also probably need to request that receipts be sent over an asynchronous connection. See the chapter on collaboration settings for details.

### **Attempt restarts**

Select this to turn on checkpoint-restart. A checkpoint is information saved to disk that is a recovery point in the event of a transport failure. The restart program uses the last saved checkpoint and starts again, ensuring no loss of data.

The checkpoint files are saved on the server under the [install directory]/common/data/http/restartable, which is normally common to all nodes in the cluster. Thus, if a transfer is interrupted and the load balancer directs the restart request to a different node, the restart file will be accessible to the new node even though it did not process the original request.

To reduce unnecessary overhead when processing small files, checkpoint files are only created for messages that are at least 100 KB in size. Also, if a restart is attempted for a message whose checkpoint file on the server is more than four hours old, the checkpoint file will be discarded and the entire message will be retransmitted.

The restart logic is used only during transport retries, such as might occur when a transfer is interrupted due to network problems. If you resubmit a message in Message Tracker, no attempt is made to perform a checkpoint-restart.

This feature only works if your partner uses Interchange or Activator and its embedded HTTP server. Do not select this option if a partner uses an external or staged HTTP server or uses a trading engine other than Interchange or Activator.

**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 256 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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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.

#### **Restrict maximum file size for this transport**

Optionally lets you specify the maximum size of files a transport can handle.

If the trading engine receives a file larger than the maximum, the file is rejected and a message is written to the events log. If received via HTTP, a 413 response also is sent and the connection is closed. A 413 message is Request Entity Too Large.

The maximum size must be expressed in bytes. Do not use commas. For instance, a kilobyte is 1024 bytes, a megabyte is 1048576 bytes, a gigabyte is 1073741824 bytes.

The smallest maximum allowed is 1000 bytes. On the opposite extreme, you can enter the largest number the field can accommodate.

This control is available only for transports used for picking up messages from integration or receiving messages from partners.

## **Web services trading maintenance**

The following topics document the fields on the maintenance pages for the web services transport used for trading messages between partners.

Maintenance for this message protocol transport is much the same as for [HTTP maintenance](#) on page 305. When the web services message protocol is set up in a partner profile, see [HTTP or HTTPS settings tab \(partner\)](#) on page 308 and [Advanced tab \(partner\)](#) on page 308 for maintenance information.

**Note:** This topic covers maintenance of the web services transport for exchanging messages between a community and partners. For maintaining the web services integration transport, see [Web services integration maintenance](#) on page 354.

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 388.

When the trading engine sends a message via HTTP, it is the HTTP client. The endpoint the trading engine is connecting to is the HTTP server.

HTTP error codes are reported in Activator log files. When you see an HTTP code in a log file, the trading engine merely is echoing the code returned by an HTTP server, firewall, proxy server or load balancer.

The response code for a successfully sent message is 200 (meaning “OK”). When a 503 code is returned, this may be due to action by the trading engine system throttle. A 503 code indicates service temporarily is unavailable. The throttle attempts to monitor the message load by looking at available memory and the depth of internal queues. When a loaded condition is detected, consumption of new work is halted. This means two things: exchange points stop polling and embedded servers return a 503 code.

For a complete list of HTTP codes, go to <http://www.ietf.org/rfc/rfc2616.txt> and see section 10 of the document.

## 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. When you export your community profile as a partner profile, the URL becomes part of your exported partner

profile. 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 path 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. When you export your community profile as a partner profile, the URL becomes part of your exported partner profile. 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 path where partners must connect to send documents.

## Advanced tab

### Receive handler config file

Specifies the WSDD file to use when unpacking a message. The default file is **default\_receive\_handler.wsdd**. The file is in [install directory]\[build number]\conf.

### Integrate SOAP body

Specifies whether to integrate the contents of the SOAP body.

**Integrate attachments**

Specifies whether to integrate the attachments of the SOAP message.

**Synchronous response generated in backend**

Tells the Web Services Protocol Receiver whether to hold open inbound HTTP connections so a synchronous response can be generated by the back-end application.

**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 strongly 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. Without backups, a message in process cannot be recovered if the server or a processing node stops or restarts. 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.

**Restrict maximum file size for this transport**

Optionally lets you specify the maximum size of files a transport can handle.

If the trading engine receives a file larger than the maximum, the file is rejected and a message is written to the events log. If received via HTTP, a 413 response also is sent and the connection is closed. A 413 message is Request Entity Too Large.

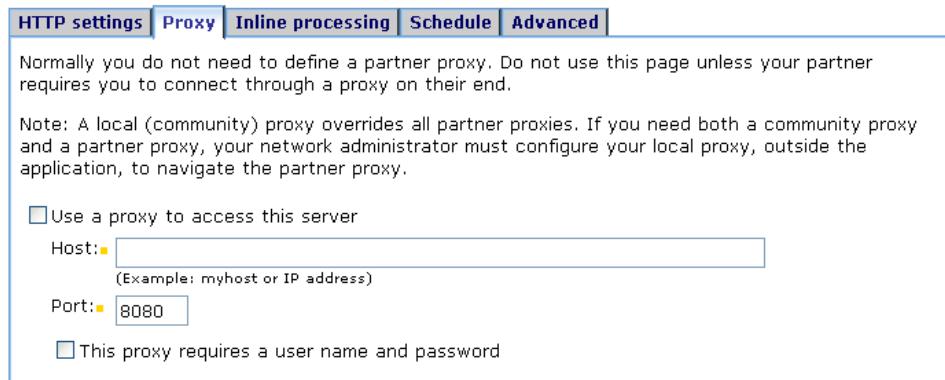
The maximum size must be expressed in bytes. Do not use commas. For instance, a kilobyte is 1024 bytes, a megabyte is 1048576 bytes, a gigabyte is 1073741824 bytes.

The smallest maximum allowed is 1000 bytes. On the opposite extreme, you can enter the largest number the field can accommodate.

This control is available only for transports used for picking up messages from integration or receiving messages from partners.

## 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.



**Figure 71. 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.

File system settings **From address** Message attributes EDI Splitter Inline processing Schedule Advanced

Use these settings 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 address

If the document is XML, use XPaths to locate the address

Document type:

Note: Selecting a document type is only for use in filling in a sample xpath below.

From XPath:

Specify the address. Always use a fixed address.

From:

Address determined by message attribute configuration.

**Figure 72. 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 address***

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 address***

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** 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.

#### **Specify the 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 368

**File system settings** | **From address** | **To address** | **Message attributes** | **EDI Splitter** | **Inline processing** | **Schedule** | **Advanced**

### 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	Selected attributes
From routing ID To routing ID Document class Content MIME type Document type Business protocol Integration ID	Add >> << Remove  Move up Move down
<a href="#">Create new attribute</a>	

### Fixed message attributes

Assign fixed values to message attributes

Name	Metadata name	Value
There are no fixed message attributes defined		

---

**Add a fixed message attribute**

Attribute name:  [Create new attribute](#)

Value:

[Add](#)

**Figure 73. 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 from partners, 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. All communities check all of these transports for outbound messages. But a community only picks up the messages that pertain to it and its partners.

Type	Location	Name	
Other (Plain text) from File system	C:\data\ediout		<a href="#">Test...</a> <a href="#">Delete</a>
Other (Plain text) from File system	C:\data\xmlout		<a href="#">Test...</a> <a href="#">Delete</a>
Other (Plain text) from File system	C:\data\binaryout		<a href="#">Test...</a> <a href="#">Delete</a>

**Figure 74. Integration pickup exchanges**

- 3** Click the name of an integration exchange to open its maintenance page.

### Change this integration pickup exchange

Transport: *File system*

File system settings | From address | To address | Message attributes | EDI Splitter | Inline processing | Schedule | Advanced

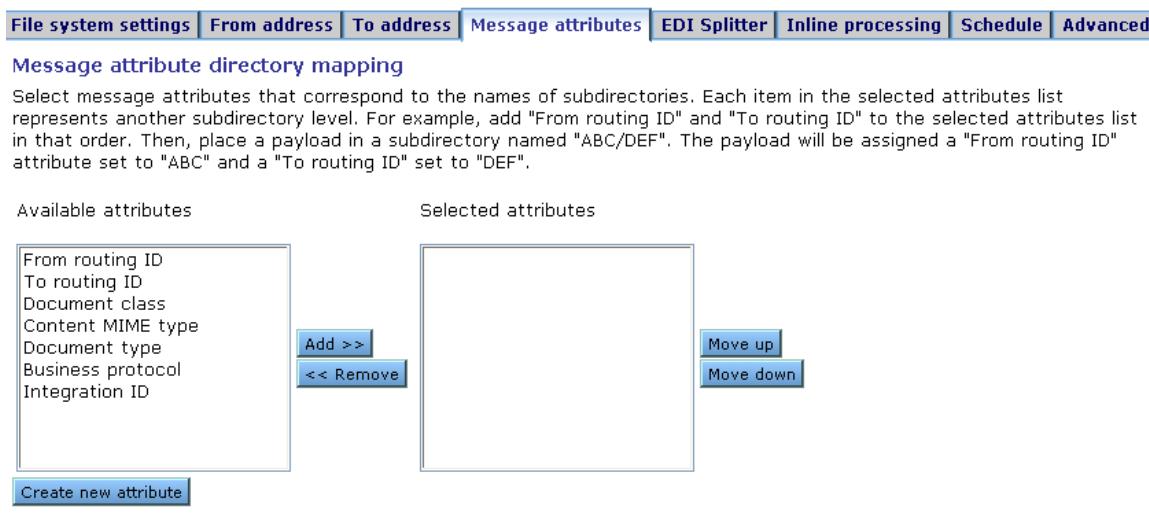
Enter the directory path. This must be a directory on the trading engine server's file system. If the directory does not exist, the system creates it. If you use the trading engine in a multiple computer cluster, make sure the directory is shared.

Directory name:  [Browse...](#)  
(Example: C:\temp or /usr/temp)

**Figure 75. 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.



**Figure 76. 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 368

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 new 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:\data1\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. All communities check all of these transports for outbound messages. But a community only picks up the messages that pertain to it and its partners.

Type	Location	Name	
Other (Plain text) from File system	C:\data\ediout		[Test...]
Other (Plain text) from File system	C:\data\xmlout		[Test...]
Other (Plain text) from File system	C:\data\binaryout		[Test...]

**Figure 77. 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 fixed message attributes defined		

**Add a fixed message attribute**

Attribute name:

Value:

**Figure 78. 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 365

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 266 or [Using fixed meta-data](#) on page 556.

If the attribute you want is not listed, do the following to add it.

Click the **Create new 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.

<a href="#">FTP settings</a>	<a href="#">Directories</a>	<a href="#">From address</a>	<a href="#">To address</a>	<a href="#">Message attributes</a>	<b>EDI Splitter</b>	<a href="#">Inline processing</a>	<a href="#">Schedule</a>	<a href="#">Adva</a>
The EDI splitter breaks apart documents containing more than one interchange into separate documents. The splitter support the X12, EDIFACT and TRADACOMS formats.								
<input type="checkbox"/> Enable the EDI splitter								

**Figure 79. 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.

## Schedule tab

The Schedule tab lets you set times for making a delivery exchange inactive while keeping the transport in an enabled state. For example, you could set a schedule to turn off the exchange for a few hours once a week to perform maintenance on a transport server.

By default exchange points are active continuously. Schedules are added by day of the week and time of day. For instance, if you select Monday 0:00 - 23:59, the exchange will be on all day every Monday. If you select Monday 8:30 - 11:30, the exchange will be on from 8:30 to 11:30 a.m. and off all other times on Mondays.

Times are expressed in 24-hour format: hh:mm or h:mm. Times are the time zone for your server. Your server's time zone is displayed in parentheses following the sentence **Times are in your server's time** on the Schedule tab.

If you schedule down times for a delivery exchange used by a community to receive messages from partners, you may want to inform partners when the transport will be inactive.

Be aware that if you want an exchange to be active most of the time but turned off only some of the time, you may need many schedules specifying the daily times you want the transport to be on or off. For example, to schedule a transport to turn off between 1 and 2 p.m. each Saturday, eight schedules are needed as follows: six daily schedules calling for the transport to run continuously Sunday through Friday and two Saturday schedules, the first specifying the transport is on from midnight to 1 p.m. and the second specifying the transport is on from 2 p.m. to midnight.

Messages in queue when a transport turns off are in suspense until the transport turns back on. For example, if a message is picked up from integration while the transport for sending to a partner is turned off, Message Tracker reports the status for the message as "scheduled production." When the transport turns on again, processing of the message continues. By the same token, retries and resends for messages are suspended while the transport is off, but pick up at the point suspended when the transport turns back on.

To use schedules, make sure message backups are enabled for the affected transports. Unless backups are enabled, messages in process when a transport turns off cannot be queued to resume processing when the transport turns on again. The check box to enable or disable backups is on the Advanced tab of each transport's maintenance page.

If you trade via the AS2 message protocol and request asynchronous receipts, your community cannot receive receipts from partners when the sending transport is turned off. To avoid this, request synchronous receipts or schedule a transport to be off when no messages are in process.

By default, this exchange point is on continuously, unless you set a schedule to enable it at specific times. To set a schedule, select the days of the week you want the exchange to be active. Type times in the 24-hour format hh:mm or h:mm. For example, 8:30 - 15:30 is 8:30 a.m. to 3:30 p.m. Times are in your server's time (Mountain Standard Time).

**Schedule**

This exchange point is always on.

**Add exchange point schedule**

Sunday	▼	0:00	-	23:59	<b>Add</b>
--------	---	------	---	-------	------------

**Figure 80. Schedule tab**

## 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.

<input type="checkbox"/>	Type	Location	Name	Delivery criteria				
<input type="checkbox"/>	FTP Other (Plain text) / File system	C:\data\ediin		( Content MIME type equals application/EDI-X12 ) OR ( Content MIME type equals application/EDIFACT ) OR ( Content MIME type equals application/EDI-consent ) OR deliver by default				
<input type="checkbox"/>	FTP Other (Plain text) / File system	C:\data\xmlin		Content MIME type equals application/xml				
<input type="checkbox"/>	FTP Other (Plain text) / File system	C:\data\binaryin		Content MIME type equals application/octet-stream				

**Change status**

**Figure 81. 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: *Other (Plain text)*, Transport: *File system*

Enable this delivery exchange  
 Name:

Make this the default delivery exchange

**File system settings** **Delivery criteria** **Message attributes** **Inline processing** **Schedule** **Advanced**

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: Content MIME type equals application/xml

		Insert		
Content MIME type	equals	application/xml	AND	OR
<input type="button" value="Delete"/>				

**Figure 82. 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.

MIME type	Description
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 215
- ◆ [Embedded HTTP transport](#) on page 219
- ◆ [Embedded transport servers](#) on page 183





# 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.

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 83. 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 84. 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](#)
- ◆ [Why use encryption and digital signatures](#) on page 383
- ◆ [The trading engine encryption method](#) on page 384
- ◆ [Encryption and signing summary](#) on page 385
- ◆ [Certificate basics](#) on page 388
- ◆ [SSL authentication](#) on page 388
- ◆ [Giving certificates to partners](#) on page 390
- ◆ [Self-signed or CA certificates](#) on page 391
- ◆ [When to get certificates](#) on page 391
- ◆ [What to do with expiring certificates](#) on page 392
- ◆ [Trusted roots](#) on page 394
- ◆ [Auto importing of intermediate and root certificates](#) on page 395

## 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 399). 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.

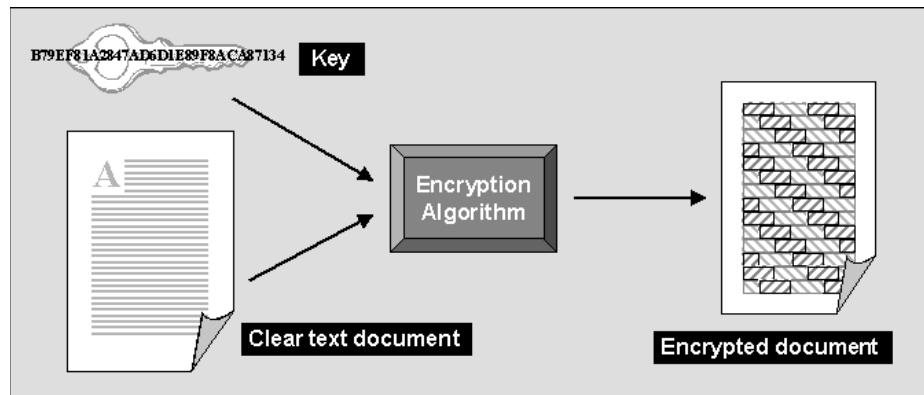


Figure 85. Encrypting a document using a key

## 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 437.
- 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 known as a PSE or personal security environment).

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 201 and [Transport maintenance](#) on page 295.

**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 Activator. 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 Axway partners**

If your trading partner also uses Interchange or Activator 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 424.

## **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 407](#) or [Import certificates for partners on page 421](#).

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.

Expiration dates for certificates are displayed in the user interface. For a community profile, click **Certificates** in the navigation graphic at the top of a community summary page to display a list of the community's certificates. The list includes the expiration days of all certificates. For a partner profile, you can view the same type of information by clicking **Certificates** at the top of a partner summary page.

The trading engine server checks at least once a day for certificates that are close to their expiration dates. A check is performed after the server is started. Thereafter, the trading engine performs a daily check. The time the check is performed depends on the value of the Interval element in the alerts.xml file, which is at [install directory]\[build number]\conf. If the interval is less than or equal to 60 minutes, the check is performed between midnight and 1 a.m., server time. If the interval is much less than 60, the check may be performed twice or more before 1 a.m. If the interval is greater than 60, the check is performed at the time past midnight equal to the interval length. For example, if the interval is 90 minutes, the check is performed at 1:30 a.m.

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.

If there are outstanding alerts for a certificate about to expire, the trading engine continues generating alerts at the interval specified in the alerts.xml file, regardless of time of day, until the certificate is replaced.

The messages about expiring certificates remain until the certificates are deleted. Figure 86 shows messages on the home page about expiring certificates.

Tasks	
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 86. 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 407

If a user certificate, see [Synchrony CSOS](#) on page 607.

If a partner certificate, import the replacement certificate the partner sends you. See [Import certificates for partners](#) on page 421.

**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.

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 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 424.
- ◆ 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 395.

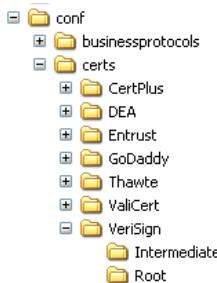
When you import a trusted root for a certificate to the trading engine, we recommend that you compare the MD5 fingerprints in both the trusted root and the certificate to verify that they match.

## 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 Activator. This is an uncommon case most users will not encounter.

To successfully trade using CA-issued certificates, 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 87 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 87. [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 [Synchrony CSOS](#) on page 607.

## Procedure

- ◆ [View certificate information](#) on page 401
- ◆ [Add a certificate](#) on page 406
- ◆ [Set up certificates for a community](#) on page 407
- ◆ [Import certificates for partners](#) on page 421
- ◆ [Export a certificate to a file](#) on page 424
- ◆ [Delete certificate](#) on page 427
- ◆ [Add a CRL](#) on page 440
- ◆ [Manage CRLs](#) on page 441

## Pages and fields

- ◆ [Certificates pages](#) on page 397
- ◆ [Community certificates page](#) on page 398
- ◆ [Partner certificates page](#) on page 400
- ◆ [Certificate field descriptions](#) on page 401

## Concepts

- ◆ [Certificate exchange messaging](#) on page 428
- ◆ [Certificate revocation lists](#) on page 437
- ◆ [Advanced CRL settings](#) on page 442
- ◆ [Analyze certificates for errors](#) on page 445

## 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.

### **Pick a certificate**

Community: *Worldwide Trading*

Personal certificates	Trusted root certificates	Trusted SSL root certificates																		
Default signing certificate is:	<input type="button" value="Worldwide2"/>																			
Default encryption certificate is:	<input type="button" value="Worldwide2"/>																			
Default certificate for SSL client authentication is:	<input type="button" value="&lt;none&gt;"/>																			
<input type="button" value="Save changes"/>																				
<table border="1"> <thead> <tr> <th>Name</th> <th>Subject name</th> <th>State</th> <th>Usage</th> <th>Expiration date</th> <th></th> </tr> </thead> <tbody> <tr> <td><a href="#">Worldwide2</a></td> <td>Worldwide Trading</td> <td>Operational</td> <td>Encryption and signing</td> <td>Mar 17, 2007 4:29:21 PM</td> <td><input type="button" value="Delete"/></td> </tr> <tr> <td><a href="#">Worldwide1</a></td> <td>Worldwide Trading</td> <td>Operational</td> <td>Encryption and signing</td> <td>Mar 17, 2007 4:27:39 PM</td> <td><input type="button" value="Delete"/></td> </tr> </tbody> </table>			Name	Subject name	State	Usage	Expiration date		<a href="#">Worldwide2</a>	Worldwide Trading	Operational	Encryption and signing	Mar 17, 2007 4:29:21 PM	<input type="button" value="Delete"/>	<a href="#">Worldwide1</a>	Worldwide Trading	Operational	Encryption and signing	Mar 17, 2007 4:27:39 PM	<input type="button" value="Delete"/>
Name	Subject name	State	Usage	Expiration date																
<a href="#">Worldwide2</a>	Worldwide Trading	Operational	Encryption and signing	Mar 17, 2007 4:29:21 PM	<input type="button" value="Delete"/>															
<a href="#">Worldwide1</a>	Worldwide Trading	Operational	Encryption and signing	Mar 17, 2007 4:27:39 PM	<input type="button" value="Delete"/>															

**Figure 88. 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 388). 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 405.

To import a trusted root, click **Add a trusted root certificate** on the certificates page and see [Import certificates for partners](#) on page 421.

### 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 399. Also see [SSL authentication](#) on page 388.

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 421.

### 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 398.

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 ▾

Save changes

Name	Subject name	State	Usage	Expiration date	
Acme Industries	Acme Industries	Operational	Encryption and signing	Nov 23, 2006 10:07:47 AM	<span style="border: 1px solid #ccc; padding: 2px;">Delete</span>

**Figure 89. 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*

**General Details Trusts**

Name:  \*

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

**Save changes**

**Or pick a task**

- [Export this certificate](#)
- [Delete this certificate](#)
- [Add a certificate](#)
- [Make this the signing certificate](#)
- [Make this the encryption certificate](#)

**Figure 90. 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 388.

### 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.

## View certificate

Community: *Worldwide Trading*

General	Details	Trusts
<b>Field</b> <b>Value</b>		
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:A8:08:D4:54:D0:A6:5C:1A:41:11:20:FA:B7:92:0A:60:F0:0E:42:90:D2:F3:EF:6E:FA:A8:F2:8B:D0:20:DD:BD:3C:D4:5B:0A:C5:3B:6C:CA:F1:4E:05:B2:2F:4F:CD: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:I02:03:01:00:01
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

**Save changes**

**Figure 91. 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.



**Figure 92. 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 the following options for adding a certificate:

- ◆ Create a self-signed certificate. See [Generate self-signed certificates on page 408](#)
- ◆ Import a certificate and private key from a file. See [Import key pair in certificate file on page 418](#).
- ◆ Import a certificate from a certificate authority. See the following topics:

[Import Entrust certificate on page 411](#)

[Import RSA Keon certificate on page 415](#)

A **partner** has the single option of importing a certificate from a file. See [Import certificates for partners on page 421](#).

If your software license allows users to have certificates, see [Import CSOS signing certificate on page 611](#).

If generating or importing a replacement certificate and you and your partners trade via the AS2 message protocol, you can use CEM to send certificates to partners. See [Certificate exchange messaging on page 428](#).

# 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.

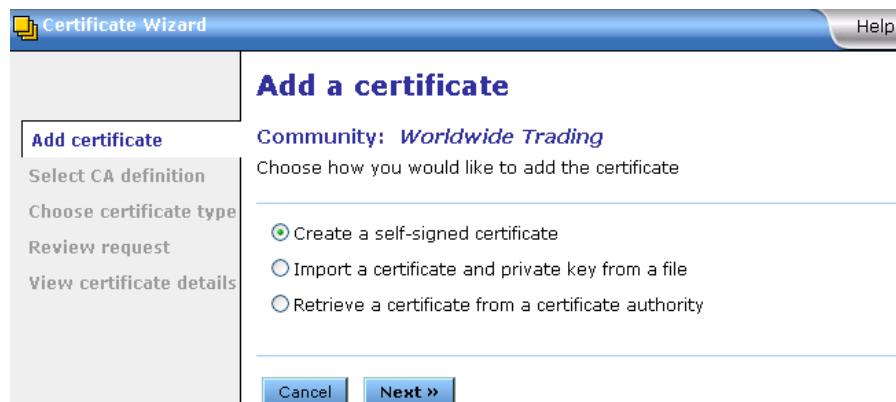
If you want to import a certificate from a third-party CA, such as VeriSign, you must obtain the certificate using your Internet browser and export it to a file before you begin this procedure. You must export the certificate to a file that contains the private key and the entire chain of trust. You 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 421.

## Steps

- 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.



**Figure 93. Certificate wizard select certificate type page**

- 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 408.

### **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 418.

### **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 411.

RSA Keon. Go to [Import RSA Keon certificate](#) on page 415.

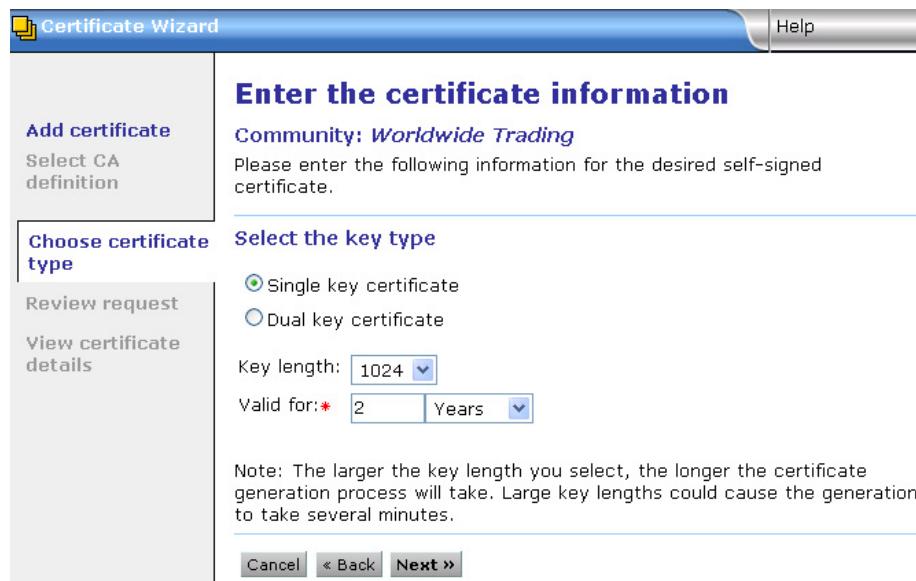
## **Generate self-signed certificates**

Use this procedure if you selected generate self-signed certificates in step 2 of [Set up certificates for a community](#) on page 407.

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.

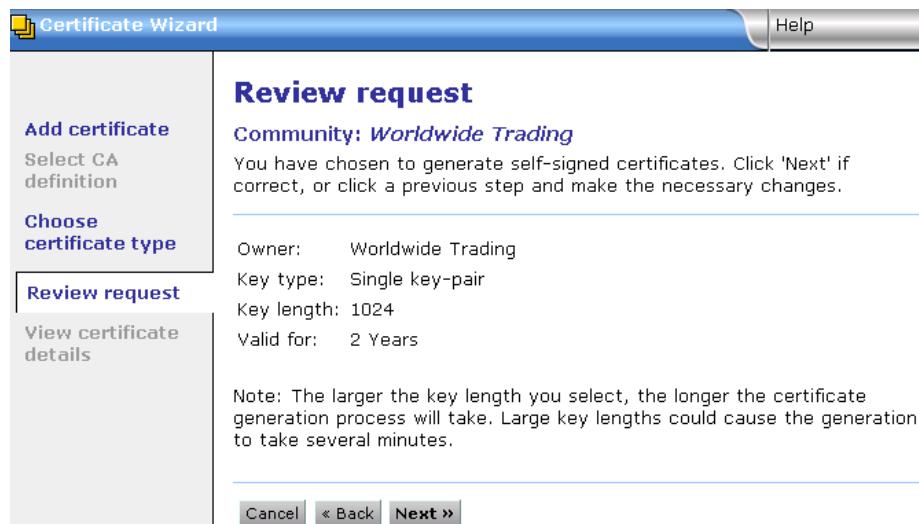
### **Steps**

- 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.



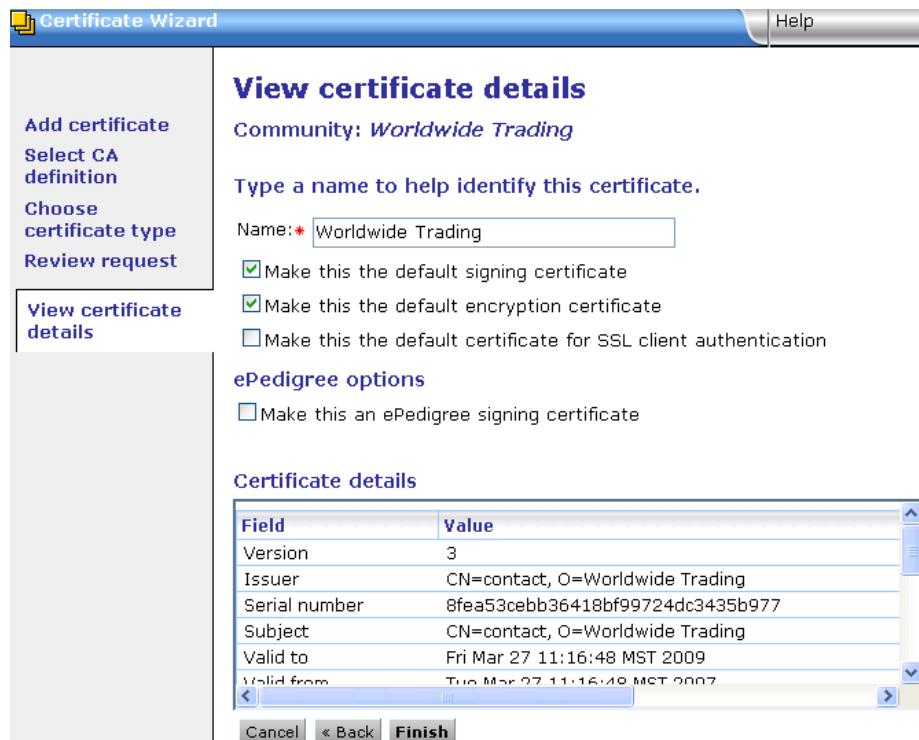
**Figure 94. Certificate wizard select certificate key type page**

- 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.
  - 3072 Very strong encryption.
  - 4096 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.



**Figure 95. 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.



**Figure 96. 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 388.

**Note:** The ePedigree certificate option displays only if your user license supports ePedigree functionality.

- 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 167.

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 424.

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 405.

## Import Entrust certificate

Use this procedure if you selected acquire Entrust certificates in step 2 of [Set up certificates for a community](#) on page 407.

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.

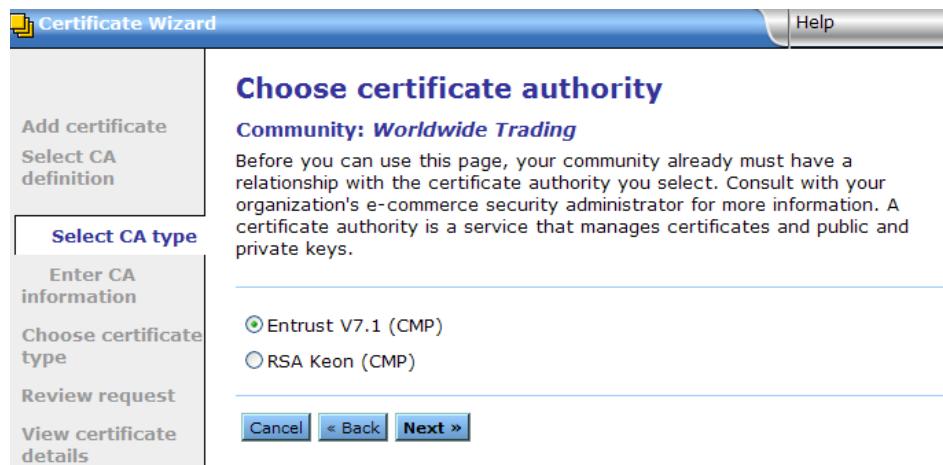
The following describes the certificate-generation process involving the trading engine and the Entrust server.

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.7.1 (CMP)**.



**Figure 97. Certificate wizard certificate authority selection page**

- 3 Click **Next** to display the certificate wizard Entrust server information page.



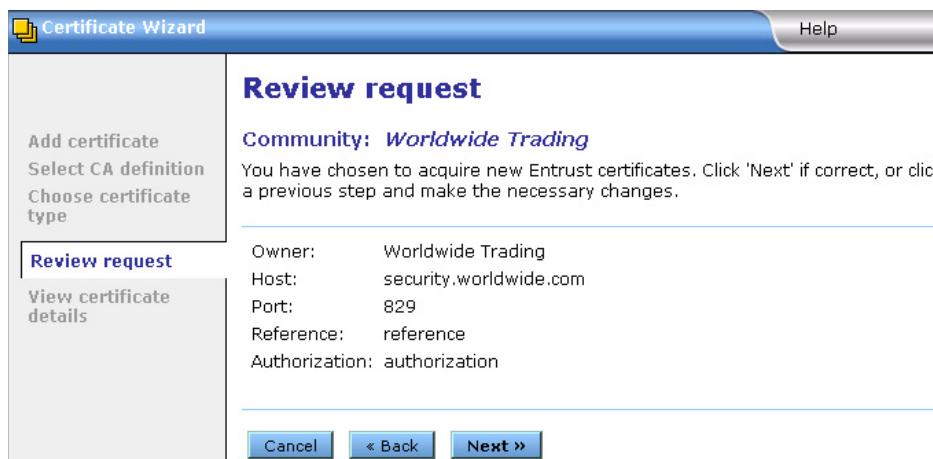
**Figure 98. Certificate wizard Entrust server information page**

- 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.



**Figure 99. 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.



**Figure 100. 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 167.

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 424.

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 405.

## 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 407.

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.

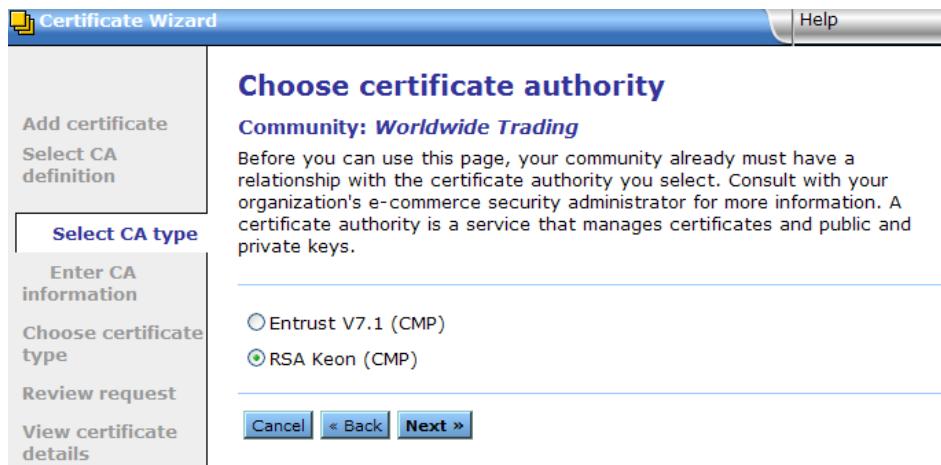


Figure 101. Certificate wizard select certificate authority page

- 2 Select **RSA Keon (CMP)** and click **Next** to display the RSA Keon host and port page.



Figure 102. 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.



**Figure 103. 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.



**Figure 104. 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 167.

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 424.

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 405.

## 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.

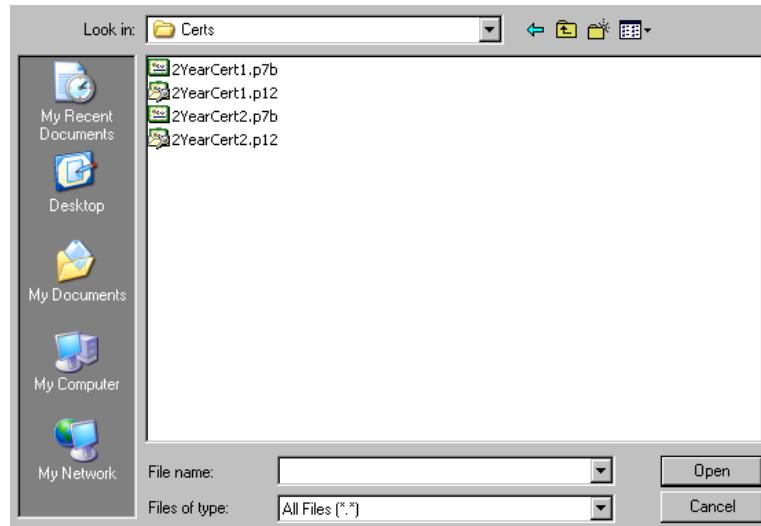
### Steps

- 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.



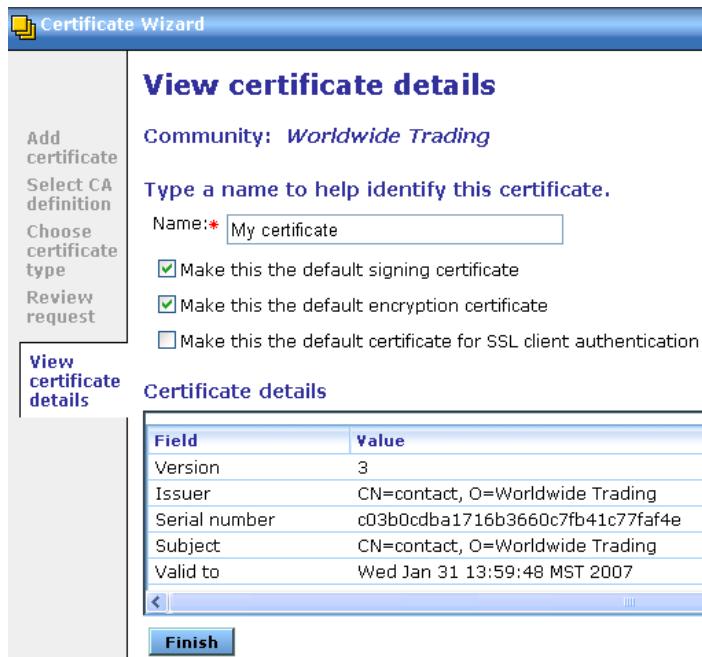
**Figure 105. Certificate wizard locate the certificate file page**

- 2 To locate the PKCS#12 file containing your certificate, click **Browse** to display the Browse dialog box.



**Figure 106. 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.



**Figure 107. 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 388.

- 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 167.

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 424.

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 405.

## 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 Axway 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.



Figure 108. Certificate wizard import certificate from a file page

- 5 Click **Next** to display the file selection page.

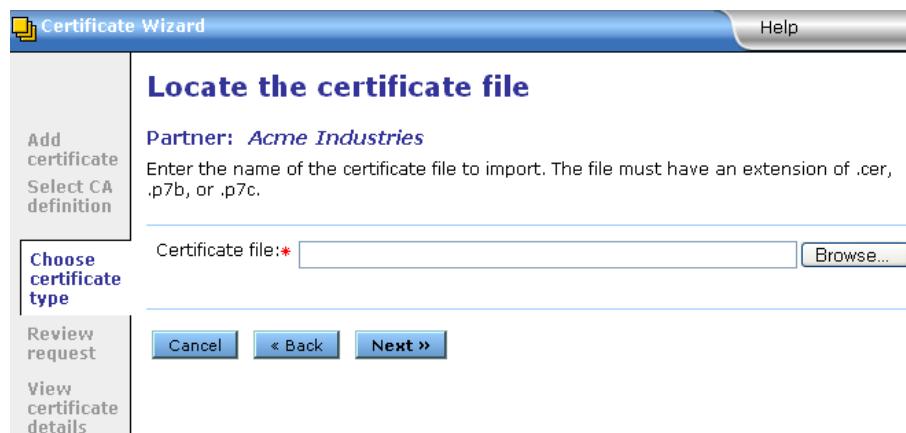
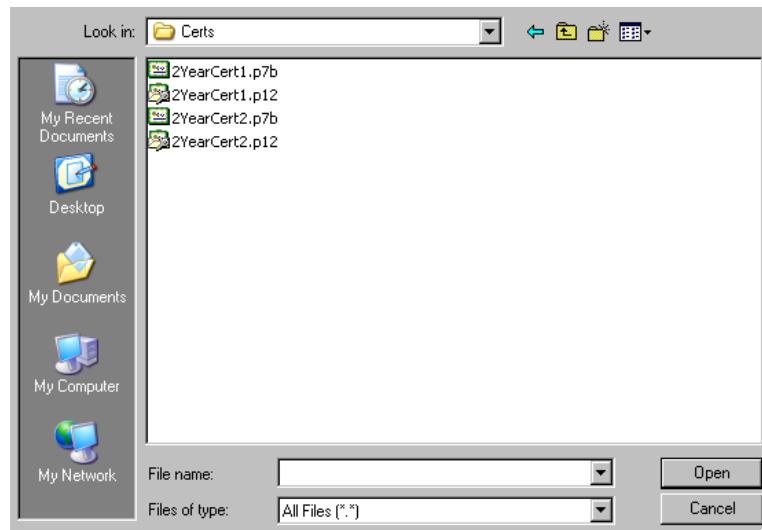


Figure 109. Certificate wizard certificate file page

- 6 Click **Browse** to open the Browse dialog box.



**Figure 110. 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.



**Figure 111. 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 388.

- 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 405.

## 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.

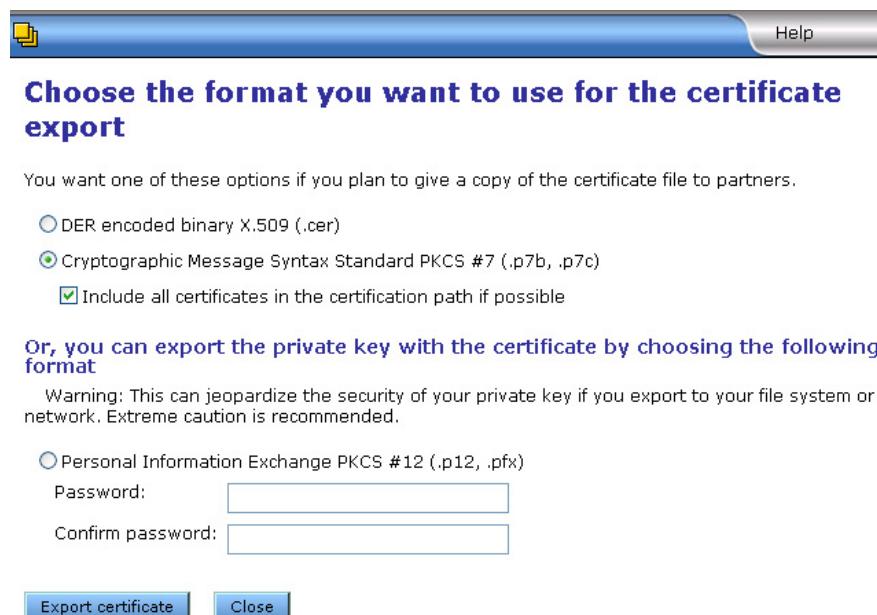
For yourself for backup purposes, you can export a community certificate that contains your private and public keys. If you do, keep this certificate in a secure place and never give it to anyone.

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 112 shows a community certificate export page. A partner certificate export page does not have the option for exporting a private key.



**Figure 112. 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 Axway 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)**

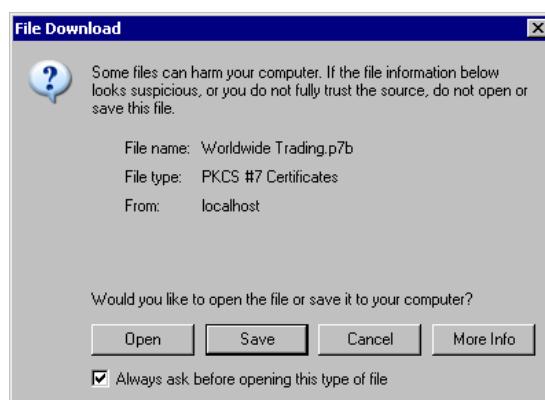
Select this option to export a file with an extension of p7c. The file can contain all the certificates needed to support trading, if more than one is required. If your partner's software supports a certificate in a PKCS#7 format, we recommend this option over DER.

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 Axway, might not support the entire certificate path in the p7c file.

**Personal Information Exchange PKCS #12 (.p12, .pfx)**

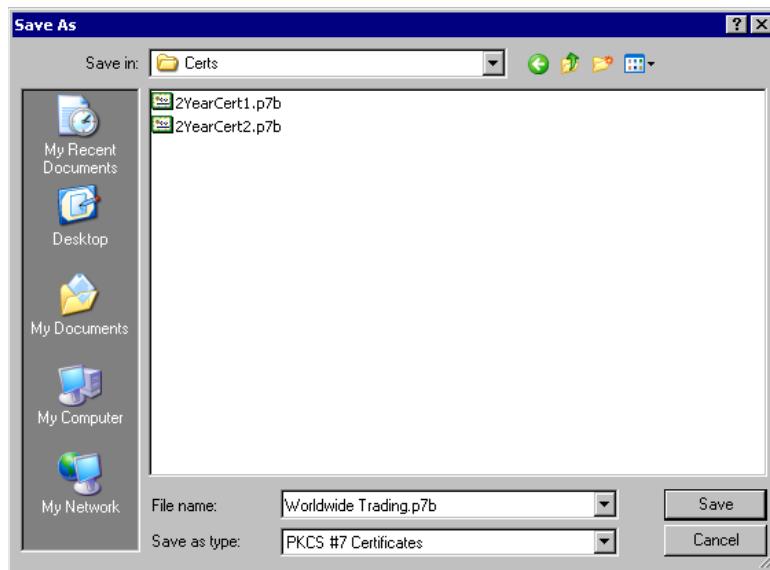
Select this option to export a community certificate containing the private key. You should do this only if you can keep the certificate in a highly secure place. This option is available only for community certificates and not partner certificates. Your user role must have permissions to export private keys.

- 6** Click **Export certificate** to display a file download dialog box.



**Figure 113. File download dialog box**

- 7** Click **Save** to display the Save As dialog box.



**Figure 114. 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 exchange messaging

Certificate exchange messaging (CEM) is a standard for EDIINT trading partners to exchange public-key certificates, replacing certificates that are about to expire, been compromised or need replacing for some other reason.

The primary purpose of CEM is to replace certificates with new ones. It is not intended as a way to distribute certificates to partners the first time. When replacing, the old certificates are used to sign CEM messages. This authenticates the identities of the senders.

CEM is a draft standard of the Internet Engineering Task Force (IETF). The IETF draft is available at the following URL:

<http://www.ietf.org/internet-drafts/draft-meadors-certificate-exchange-06.txt>

The IETF standard encompasses certificate exchanges between partners who use the AS1, AS2 and AS3 message protocols. Activator supports CEM for AS1, AS2, AS3 and also the Secure File message protocol. The protocols are supported for Activator communities and partners who use Activator or a third-party interoperable trading engine.

The exchange of replacement ePedigree signing certificates is not supported.

The following topics describe how the trading engine implements CEM, but this documentation is not a substitute for reading and understanding the IETF CEM document.

## ***Types of CEM messages***

CEM messages exchanged between trading partners are XML documents. There are two kinds of messages: CEM request and CEM response.

CEM messages always concern replacement end-entity certificates. The CEM standard presumes partners already have exchanged public-key certificates, either to begin or continue a trading relationship.

Before sending a CEM request, a community obtains a replacement certificate and public-private key pair. In the certificate wizard, the community can generate a self-signed certificate or import a CA certificate.

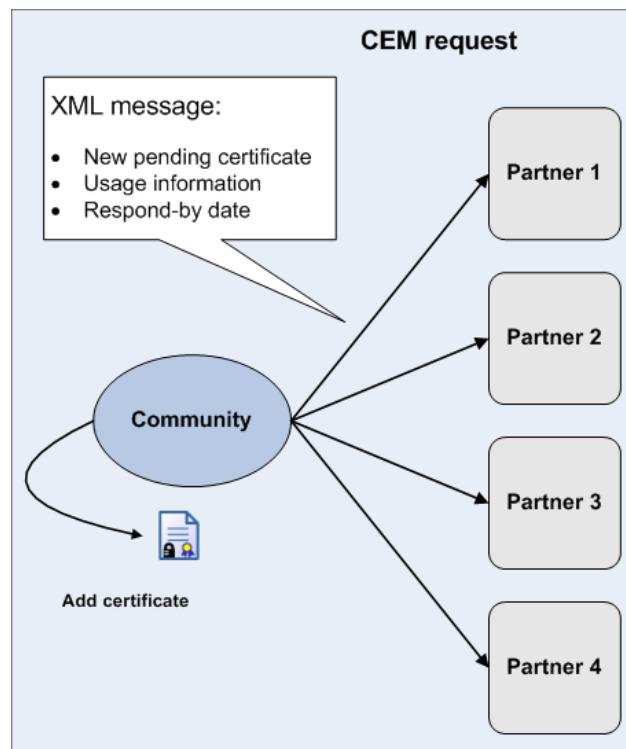
The following describes how the trading engine handles request and response messages.

## CEM request

A CEM request (Figure 115) is an XML message sent by a community to one or more partners. The message asks partners to begin using the included public-key certificate in the trading relationship.

The message includes a respond-by date. Partners are asked to tell the community by that date whether the new certificate has been accepted.

The message also states the use for the new certificate (for example, signing, encryption, SSL).



**Figure 115. CEM request**

## CEM response

A CEM response (Figure 116) is an XML message sent by one or more partners. The message tells the community whether the partner has accepted the certificate in the CEM request.

If all partners accept the certificate before or on the respond-by date, the community implements the certificate. If even a single partner rejects the certificate, the community does not implement the certificate.

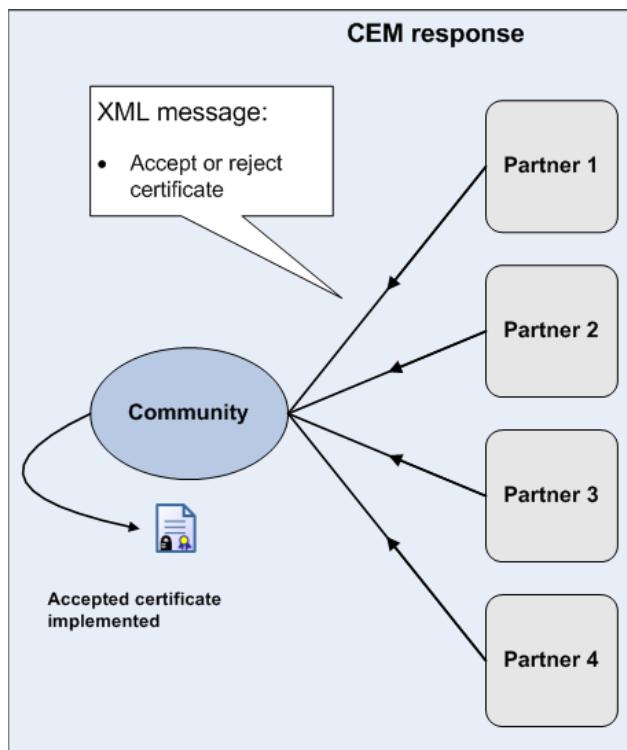


Figure 116. CEM response

## ***Types of certificates in CEM requests***

Certificates used for the ordinary purposes of signing, encryption and SSL can be sent in CEM requests for acceptance by partners. There are some variations in the way the trading engine handles certificates slated for different uses that you should know about.

### **Encryption**

When a community sends a CEM request for a new encryption certificate, the community must be prepared immediately to receive messages from partners encrypted with the new certificate and the old certificate the new certificate is intended to replace. The trading engine handles this without user intervention. Upon receiving an encrypted message, the trading engine determines the public key used to encrypt and uses the proper private key from its key store to decrypt.

### **Signing and SSL**

A community uses a signing certificate to sign outbound messages. SSL certificates are used to authenticate a client or server.

When a community sends a CEM request for a new signing or SSL certificate, the community does not implement the certificate unless one of the following conditions is met:

- The respond-by date in the CEM request has passed and no partners have sent a CEM response rejecting the certificate. This means absent any responses — positive or negative — by the respond-by date, the community implements the certificate when the respond-by date has passed.
- On or before the respond-by date, all partners have returned CEM responses accepting the certificate. If all partners respond positively before the respond-by date, the community implements the certificate when the final positive respond has been received.

If only one partner sends a CEM response rejecting the certificate, the community will not implement the certificate.

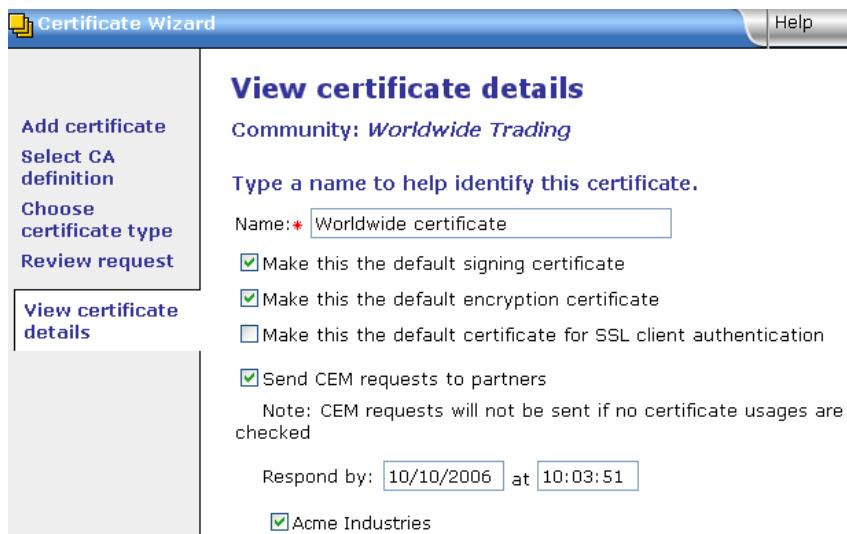
Moreover, upon accepting a new signing certificate, a partner must be able to accept messages from the community signed by either the new certificate or the old certificate the new certificate is intended to replace.

## **Send a CEM request**

Use this procedure to send a CEM request to one or more partners.

### **Steps**

- 1 In the user interface, launch the certificate wizard to generate a self-signed certificate or import a CA certificate. See [Add a certificate](#) on page 406.
- 2 On the View certificate details page in the certificate wizard, select the use (signing, encryption, SSL).
- 3 Select **Send CEM requests to partners**. Select the respond-by date and the partners to whom you want to send the CEM request.



**Figure 117. View certificate details page**

- 4 Click **Finish** to complete the process of generating or importing the certificate. This action also sends the CEM request to the selected partners.
- 5 Open the Sent CEM requests page to view the status of CEM requests. See [Sent CEM requests page](#) on page 435.

## Track CEM requests

There are two pages in the user interface for monitoring CEM messages.

Page	Description
<a href="#">Received CEM requests page</a>	The Received CEM requests page provides information about the CEM requests your community has received from partners. This includes the date the message was received, the name of the partner who sent the message and the number of public-key certificates in the message.
<a href="#">Sent CEM requests page</a>	The Sent CEM requests page provides information about the CEM requests your community has sent to partners. This includes the sent and respond-by dates, the number of partners to whom the request was sent and the number of public-key certificates in the message.

More details about these pages are provided in the following topics.

## Received CEM requests page

The Received CEM requests page (Figure 118) provides information about the CEM requests your community has received from partners. This includes the date the message was received, the name of the partner who sent the message and the number of public-key certificates in the message.

### Received CEM requests

Received CEM requests are messages from trading partners. These messages present certificates that the partner intends to begin using in the trading relationship.

You can choose to either automatically or manually accept CEM requests.

- Let the system automatically accept all signed CEM requests
- All CEM requests must be manually accepted or rejected

[Save changes](#)

	Date	Partner	Certificates	
<a href="#">Request details</a>	Aug 2, 2006 2:54:58 PM	Acme Industries	1 total, 0 accepted, 0 rejected	<a href="#">Message details</a>
<a href="#">Request details</a>	Aug 11, 2006 1:39:31 PM	Acme Industries	1 total, 0 accepted, 0 rejected	<a href="#">Message details</a>

**Figure 118. Received CEM requests page**

To open the page, select **Trading configuration** on the top toolbar. On the Pick a community page, click **Manage received CEM requests** in the task list at the bottom of the page.

You can use the Received CEM requests page to determine how you want to manage processing of inbound requests:

- Select **Let the system automatically accept all signed CEM requests** if you want your community to accept all replacement certificates from partners. The trading engine approves each certificate upon receipt and sends an acceptance CEM response to the partner who sent the message. Approval is contingent on whether the CEM request was signed with a certificate associated with the partner's profile.
- Select **All CEM requests must be manually accepted or rejected** if you want to review each CEM message and determine whether to accept or reject the certificate. The trading engine sends a CEM response once you have decided.

Regardless whether you choose the automatic or manual option, the trading engine performs the following internal checks before accepting a certificate:

- 1 Does the trading engine have the complete certificate chain for the certificate in the CEM request? If a self-signed certificate, the CEM request should contain the complete certificate chain.
- 2 Does the CEM request XML document contain the issuer name and serial number of the certificate in the request? The trading engine makes sure the issuer name and serial number are the same in the XML document and in the certificate within the XML document.
- 3 Does the certificate support the requested usage of signing, encryption or SSL? For example, if the CEM request says the certificate is to be used for signing, the trading engine makes sure the key usage extension, if any, in the certificate actually supports digital signatures.

On the Received CEM requests page, a **Message details** link displays for each message. Click the link to open a message details page in [Message Tracker](#). CEM messages are sent and received like any other messages exchanged between partners.

After a CEM request has been accepted or rejected, a **Delete** button appears on the Received CEM requests page. This lets you clear the page of old CEM request records.

On the Received CEM requests page you can click **Request details** to open the Received CEM request details page (Figure 121).

### Received CEM request details

Request date: Aug 11, 2006 1:39:31 PM

Request partner: [Acme Industries](#)

The CEM request has specified that the listed certificates be placed into the trading relationship. You must either accept or reject them.

Name	Issuer	Respond by date	Usage	Response	
<a href="#">Acme Industries</a>	Acme Industries	Aug 18, 2006 1:39:05 PM	Encryption, Signing	Pending	<a href="#">Accept</a> <a href="#">Reject</a>

**Figure 119. Received CEM request details page**

The Received CEM request details page shows details about the CEM request, including the name of the partner, the requested respond-by date and the usage for the replacement certificate.

The name of the certificate is displayed in the Name column. Click the name to open a View certificate page, which contains details about the public-key certificate (see [Certificate field descriptions](#) on page 401).

If you selected the manual option on the [Received CEM requests page](#), determine whether to accept the certificate and click **Accept** or **Reject** to send a CEM response to the partner who sent the request. The Response column updates with a message reflecting whether you accepted or rejected the certificate.

If you reject a certificate, you can type a reason. The reason displays on the Received CEM request details page and is included in the CEM response to the partner who sent the CEM request.

After accepting or rejecting the request, you can click the link in the Response column to open a details page in [Message Tracker](#) for the CEM response message you sent to the partner who sent the CEM request.

## Sent CEM requests page

The Sent CEM requests page (Figure 120) provides information about the CEM requests your community has sent to partners. This includes the sent and respond-by dates, the number of partners to whom the request was sent and the number of public-key certificates in the message.

### Sent CEM requests

Sent CEM requests are messages from a community to its trading partners. These messages present certificates that the community intends to begin using in the trading relationship.

	Sent date	From	Respond by date	Recipients	Certificates	
<a href="#">Request details</a>	Aug 2, 2006 1:05:11 PM	Worldwide Trading	Aug 2, 2006 1:34:17 PM	1	1	<a href="#">Delete</a>
<a href="#">Request details</a>	Aug 2, 2006 3:07:22 PM	Worldwide Trading	Aug 9, 2006 3:07:06 PM	1	1	<a href="#">Delete</a>
<a href="#">Request details</a>	Aug 2, 2006 3:14:13 PM	Worldwide Trading	Aug 9, 2006 3:13:59 PM	1	1	<a href="#">Delete</a>
<a href="#">Request details</a>	Aug 11, 2006 9:39:42 AM	Worldwide Trading	Aug 18, 2006 9:37:36 AM	1	1	<a href="#">Delete</a>

Or pick a task

[Manage trading configuration](#)

**Figure 120. Sent CEM requests page**

To open the page, select **Trading configuration** on the top toolbar. On the Pick a community page, click **Manage sent CEM requests** in the task list at the bottom of the page.

After the respond-by date for a request has passed or has been canceled, a **Delete** button displays. You can click it to delete the record from the page. This only clears the display of the specific request; deleting does not affect the certificate.

**Note:** A respond-by date is canceled either when all partners who received the CEM request have rejected the certificate or when the request is canceled on the Sent CEM request details page.

On the Sent CEM requests page you can click **Request details** to open the Sent CEM request details page (Figure 121).

### Sent CEM request details

Sent date: Aug 11, 2006 9:39:42 AM  
From: [Worldwide Trading](#)  
Respond by date: Aug 18, 2006 9:37:36 AM

The sent CEM request has specified that these certificates be placed into the trading relationship.

Name	Issuer	Usage	Installed method	Recipients	
<a href="#">Worldwide Trading</a>	Worldwide Trading	Encryption, Signing		1 total, 0 accepted, 0 rejected	<a href="#">Install</a> <a href="#">Cancel</a>

The sent CEM request has been sent to the trading partners in each of the tables below.

To: [Acme Industries](#)

[Message details](#)

Issuer	Usage	Response
Worldwide Trading	Encryption, Signing	Pending

**Figure 121. Sent CEM request details page**

The Sent CEM request details page shows details about the CEM request, including the number of partners who have accepted or rejected the certificate.

The name of the certificate is displayed in the Name column in the top table. Click the name to open a View certificate page, which contains details about the public-key certificate (see [Certificate field descriptions](#) on page 401).

On the Sent CEM requests page you can also:

- Click **Install** to immediately implement the certificate before the respond-by date or despite rejection by one or more partners.
- Click **Cancel** to stop the trading engine from implementing the certificate after the respond-by date has passed. This forestalls implementation only in the absence of responses from partners. If all partners accept the certificate, the trading engine will implement the certificate regardless of the canceled respond-by date.

A **Message details** link displays for every partner to whom you sent a CEM request. Click the link to open a message details page in [Message Tracker](#). CEM messages are sent and received like any other messages exchanged between partners.

# 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 selecting the check box for **Automatically retrieve CRLs** on the CRL usage and retrieval configuration page (see [Advanced CRL settings](#) on page 442). 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.

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 check box for **Automatically retrieve CRLs** is selected on the CRL usage and retrieval configuration page 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 check box for **Use expired CRLs** is selected on the CRL usage and retrieval configuration page, 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 check box for **Use expired CRLs** is selected on the CRL usage and retrieval configuration page, 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 check box for **Require CRLs** is selected on the CRL usage and retrieval configuration page, 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 404. 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 437.

### Steps

- On the system management page, click **Manage CRLs** to display the CRL list page. The page lists imported CRLs, if any.

#### Pick a CRL

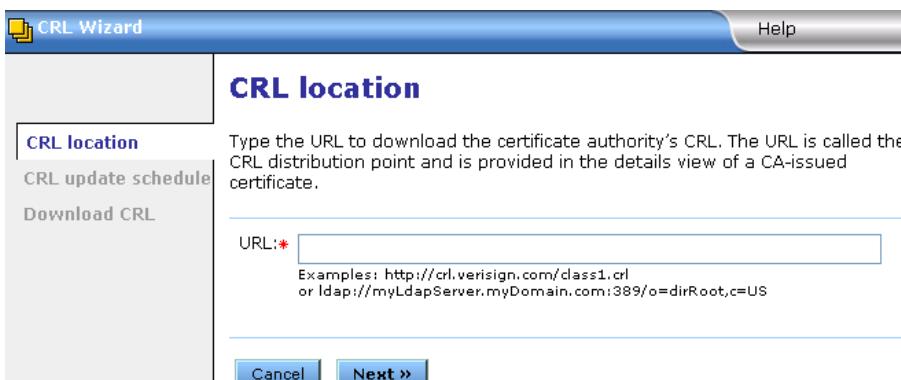
URL	Last update	Update schedule	Status
There are no CRLs defined. <a href="#">Add a CRL</a>			

Or pick a task

[Add a CRL](#)

**Figure 122. Certificate revocation list page**

- Click **Add a CRL** to launch the CRL wizard.

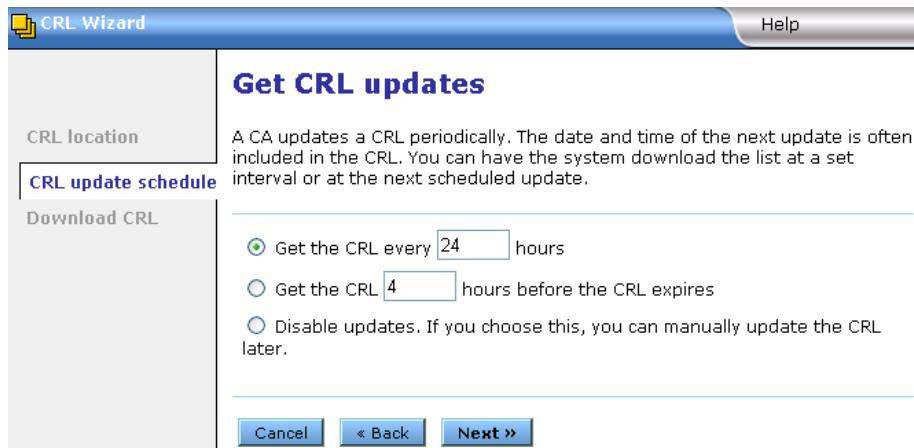


**Figure 123. CRL wizard CRL location page**

- Type the URL to download the CRL. See [Obtaining CRL access information](#) on page 439.

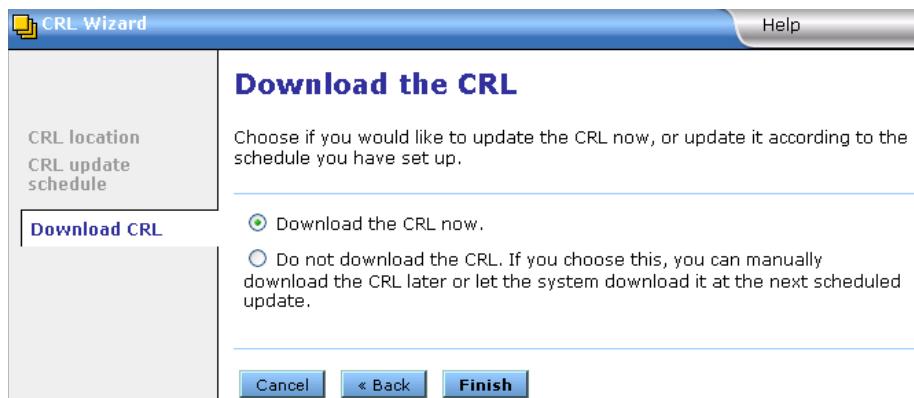
If you must connect through a proxy server to retrieve a CRL, see the proxy server fields described in [Advanced CRL settings](#) on page 442.

- Click **Next** to display the get CRL updates page.



**Figure 124.** 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.



**Figure 125.** 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 URL 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

The screenshot shows a web-based configuration interface for managing a Certificate Revocation List (CRL). At the top, there are two tabs: "Update schedule" (selected) and "CRL details". The "Update schedule" tab contains fields for the CRL URL (set to "http://crl.verisign.com/class1.crl") and update frequency (set to "Get the CRL every 24 hours"). Below these are three options: "Get the CRL 24 hours before the CRL expires", "Disable updates", and a note about manual updates. A status section at the bottom displays the last update attempt (Oct 12, 2005 11:30:07 AM), last update success (Oct 12, 2005 11:30:07 AM), and current status (Success). At the bottom of the page are "Save changes" and "Update now" buttons, and a link to "Or pick a task" followed by a checkbox for "Delete this CRL".

**Figure 126. Manage CRL page**

To delete a CRL, click **Delete** to the right of a CRL 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 fields on a page in the user interface.

For most users, the default settings are sufficient. But users with special needs may want to change some fields. When the default values are used and no current CRLs are in [install directory]\common\conf\crls, CRL checking does not occur.

To open the CRL page, select **System management** on the toolbar and click the **Manage CRLs** link. Then click **Configure CRL usage and retrieval** to open the CRL usage and retrieval configuration page.

## CRL usage and retrieval configuration

- Require CRLs
- Use expired CRLs
- Automatically retrieve CRLs
- Use proxy to retrieve CRLs over HTTP

**Save changes**

**Figure 127. CRL usage and configuration page**

**Note:** Before version 5.5, advanced CRL tuning was controlled by properties in the crossworks.properties file. Although this file remains in [install directory]\[build number]\conf, using the user interface page to change settings is recommended. The crossworks.properties file contains messages about properties whose values should be left as-is or can be changed.

The following describes the fields on the CRL usage and configuration page. These values are stored in the database.

### Require CRLs

Select 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.

When Require CRLs is turned off and a CRL cannot be found for a certificate, the certificate is presumed valid and certificate path validation continues.

When Require CRLs is turned on, 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 selected, 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.

In most cases, Require CRLs is turned off by default.

However, if you are licensed to perform CSOS processing, Require CRLs is on by default because CRL checking is presumed for CSOS.

### Use expired CRLs

Select 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.

Clear the check box to use only CRLs that have not expired.

Use expired CRLs is turned off by default.

### Automatically retrieve CRLs

Indicates whether CRLs should be automatically retrieved using the information in certificates' CRL distribution points extension.

When selected 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 **Require CRLs** also is selected.

In most cases, Automatically retrieve CRLs is turned off by default.

However, if you are licensed to perform CSOS processing, Automatically retrieve CRLs is on by default because CRL checking is presumed for CSOS.

### Use proxy to retrieve CRLs over HTTP

Select if you must connect through a proxy server to retrieve a CRL. If selected, complete the related fields.

#### *Host*

The name or IP address of the proxy server to use when retrieving CRLs via HTTP.

#### *Port*

The port number of the proxy server to use when retrieving CRLs via HTTP. If you leave this field blank, the port defaults to 80.

***This proxy requires a user name and password***

Select if a user name and password are required to connect to the proxy server. Type the authentication information in the user name and password fields.

# Analyze certificates for errors

An X.509 certificate-scanning command-line tool is available to report issues related to public-key certificates. The tool is **certScan** and is found in [install directory]\[build number]\tools.

This tool is for scanning files of public-key certificates. Those are files with extensions of .cer, .crt, .der, .p7b and .p7c. It cannot scan certificates with private keys, meaning files with extensions of .p12 and .pfx.

One use for this tool is to scan certificates from partners before importing the certificate files to the trading engine. This may be advisable if you have a partner who has provided unreliable certificates that adversely affected message trading.

The tool can scan a single certificate file or a directory of certificate files.

Run the tool from the tools directory. The format is:

```
certScan [file or directory]
```

where **file** is the path to a single certificate file and **directory** is the path to a directory containing multiple certificate files.

Each certificate file is presumed to contain one or more certificates. When a file contains more than one certificate, it is assumed the multiple certificates form a chain from end-entity certificate to CA root certificate.

When a directory is specified, the tool finds all certificate files in that directory and all subdirectories, recursively.

The tool displays the following data:

- ◆ **Info.** An interesting characteristic of the certificate that does not violate any standards and does not affect the certificate's performance in the trading engine.
- ◆ **Warning.** An anomaly in the certificate that violates RFC 3280, but does not affect the certificate's performance in the trading engine.

- ◆ **Error.** A serious problem with the certificate that may violate RFC 3280, but would cause the certificate not to function properly in the trading engine.

RFC 3280 is the standard for X.509 certificates of the Internet Engineering Task Force. A copy of RFC 3280 is at:

<http://www.ietf.org/rfc/rfc3280.txt>



# 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 481).

## Concepts

- [Hierarchy of outbound settings](#) on page 447
- [Community collaboration settings](#) on page 474
- [Partner collaboration settings](#) on page 477

## Procedure

- [View settings between partners](#) on page 448
- [View or change default settings](#) on page 450

## Pages and fields

- [Default collaboration settings](#) on page 451

## 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 128 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.

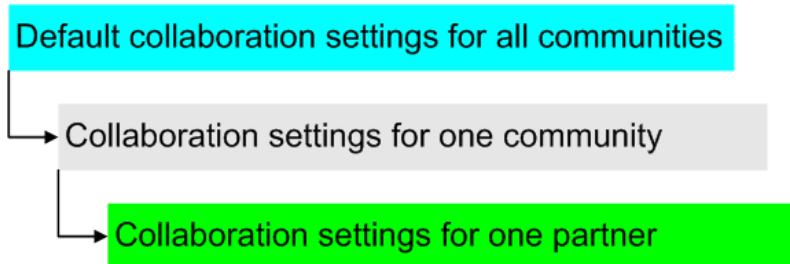


Figure 128. Hierarchy of collaboration settings

## 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:<sup>\*</sup>

Messages sent to:<sup>\*</sup>

You must supply the two parties trading above to see the settings used when sending messages.

**Figure 129. 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:*	<input type="text" value="Worldwide Trading"/>	<input type="button" value="Pick..."/>
Messages sent to:*	<input type="text" value="Acme Industries"/>	<input type="button" value="Pick..."/>

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 130. 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 447.

## 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 451 for descriptions of the fields on each tab.

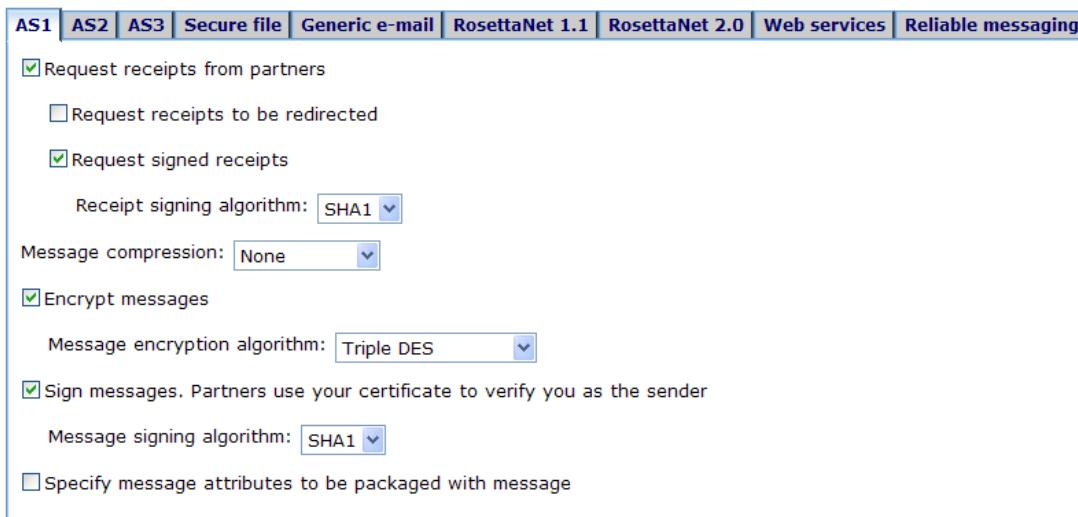
# Default collaboration settings

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 450.

## 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 131 displays the default AS1 settings tab.



**Figure 131. 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 Activator or who use an EDIINT-compliant trading engine other than Activator.

**MIME/GZIP** is for trading with partners who use versions of Activator earlier than 4.2. You also can select this if your partner uses a trading engine other than 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***

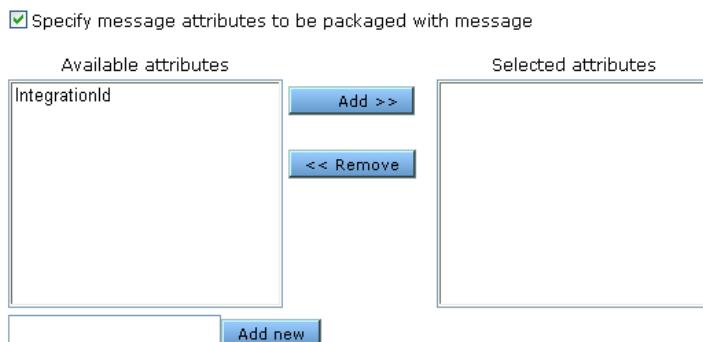
The algorithm for creating a hash of the unencrypted message. This hash is a number that is encrypted with your community's private key. It is decrypted by the partner using the community's public key. The partner rehashes the decrypted message and compares the result with the hash that came with the 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 132. 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 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 133 displays the default AS2 settings tab.

The screenshot shows the 'Default AS2 settings' configuration page. At the top, there is a navigation bar with tabs: AS1, AS2, AS3, Secure file, Generic e-mail, RosettaNet 1.1, RosettaNet 2.0, Web services, and Reliable messaging. The AS2 tab is currently selected.

Below the tabs, there are several configuration options:

- Request receipts from partners
  - Request receipts be sent over a synchronous connection
  - Request receipts be sent over an asynchronous connection
- Disposition notification URL: [Text input field] (Example: http://somehost.com:4080/exchange/thepath)
- 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 133. 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 Activator or who use an EDIINT-compliant trading engine other than Activator.

**MIME/GZIP** is for trading with partners who use versions of Activator earlier than 4.2. You also can select this if your partner uses a trading engine other than 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**

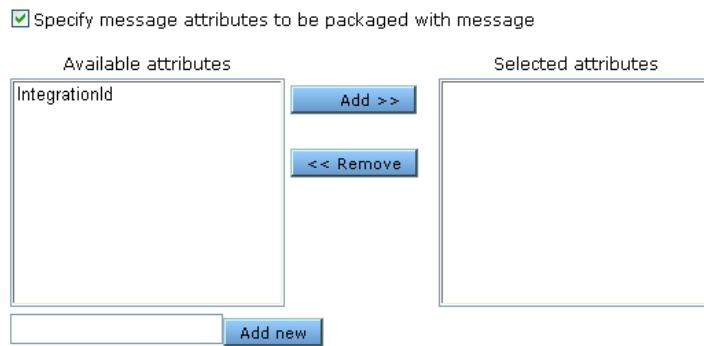
Select this check box to override the disposition notification URL for receipts that your partner set in the message your community received. When selected, the partner's disposition notification URL in the MIME header of the inbound message is ignored, and your community sends the receipt via the first 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 134. 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 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 135 displays the default AS3 settings tab.

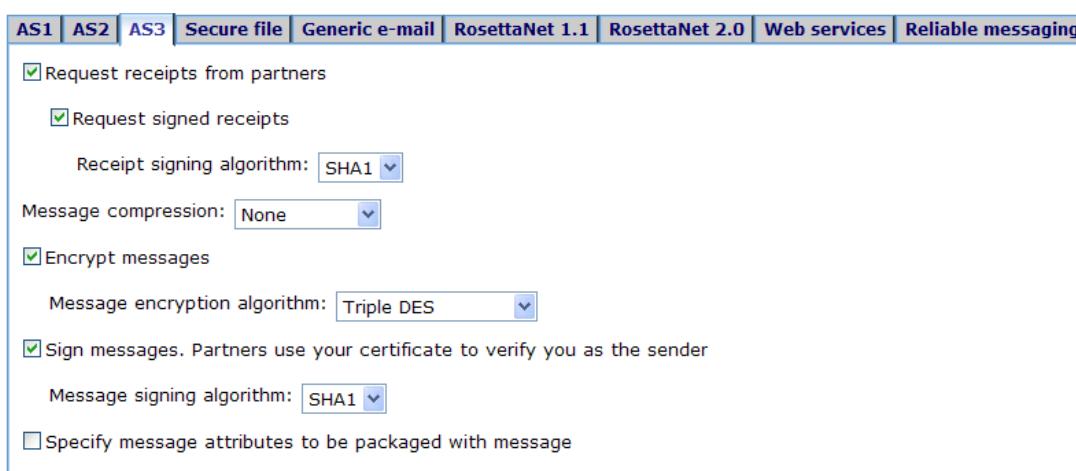


Figure 135. Default AS3 settings

### **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 Activator or who use an EDIINT-compliant trading engine other than Activator.

**MIME/GZIP** is for trading with partners who use versions of Activator earlier than 4.2. You also can select this if your partner uses a trading engine other than 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***

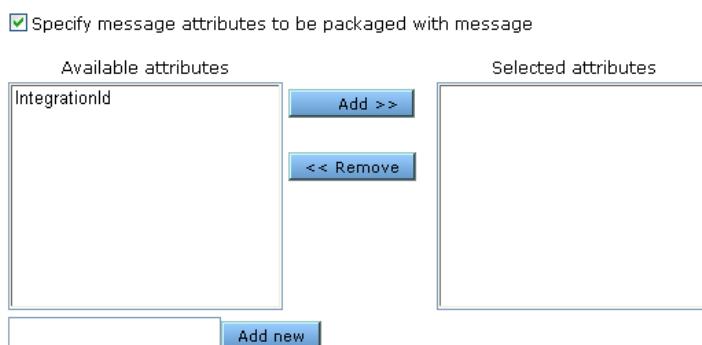
The algorithm for creating a hash of the unencrypted message. This hash is a number that is encrypted with your community's private key. It is decrypted by the partner using the community's public key. The partner rehashes the decrypted message and compares the result with the hash that came with the 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 136. 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.

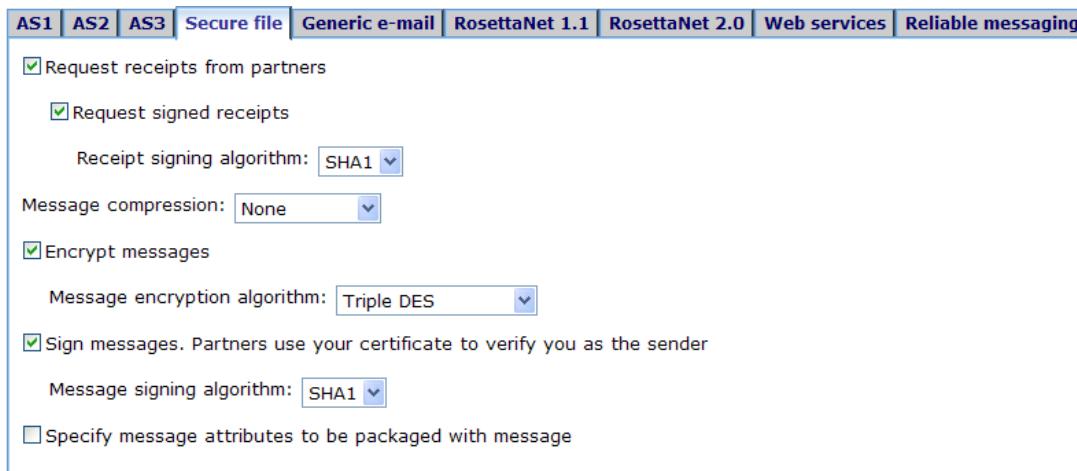
Upon receiving and unpackaging a message with extra meta-data, a partner who uses Activator 5.3.2 or later can use the added-meta-data for post processing, inline processing or file system integration. Inform the partner of the meta-data elements you add to outbound messages.

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 137 displays the default secure file settings tab.



**Figure 137. 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 Activator or who use an EDIINT-compliant trading engine other than Activator.

**MIME/GZIP** is for trading with partners who use versions of Activator earlier than 4.2. You also can select this if your partner uses a trading engine other than 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*

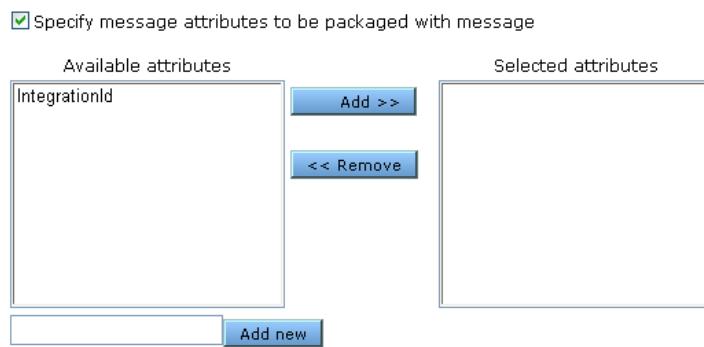
The algorithm for creating a hash of the unencrypted message. This hash is a number that is encrypted with your community's private key. It is decrypted by the partner using the community's public key. The partner rehashes the decrypted message and compares the result with the hash that came with the 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 138. 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 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

## **Generic e-mail default settings**

The generic e-mail tab shows default outbound message settings for the Generic 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 139 displays the default generic e-mail settings tab.



**Figure 139. Default generic 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 unsigned.

### 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 140 displays the default RosettaNet 1.1 settings tab.



Figure 140. 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***

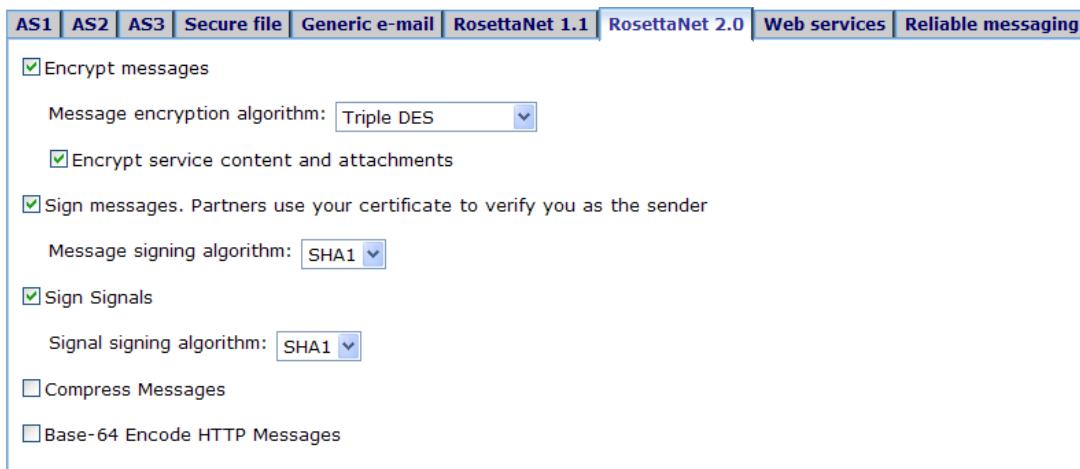
If you choose to sign signals, the signing algorithm to use.

**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 141 displays the default RosettaNet 2.0 settings tab.



**Figure 141. 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.

## Web services default settings

The Web services tab shows default outbound message settings for the web services message protocol. The settings affect all outbound documents for all communities, unless superseded by community- or partner-level settings. Figure 142 displays the default settings on the Web services tab.

The default settings enable trading of secure XML payloads between two instances of Interchange or Activator. The default configuration is somewhat arbitrary, but useful to get two instances of Interchange or Activator trading quickly. If you want a different configuration, engage a professional services consultant, obtain the optional Software Development Kit or both.

The screenshot shows the 'Web services' tab selected in a navigation bar. The configuration page includes fields for 'Send handler config file' (set to 'default\_send\_handler.wsdd'), 'Payload location' (set to 'SOAP body'), and 'SOAP action'. A checked checkbox for 'Encrypt messages' leads to a detailed configuration section. This section includes a dropdown for 'Message encryption algorithm' (set to 'Triple DES'), and checkboxes for 'Encrypt attachments' (checked), 'Encrypt SOAP body' (checked), and buttons for 'XPaths to encrypt' and 'ID refs to encrypt', both with 'Add' buttons. Another checked checkbox for 'Sign messages' leads to another configuration section with checkboxes for 'Sign attachments' (checked), 'Sign SOAP body' (checked), and buttons for 'XPaths to sign' and 'ID refs to sign', both with 'Add' buttons.

**Figure 142. Default web services settings**

### Send handler config file

Name of the Web Service Deployment Descriptor (WSDD) file in [install directory]\[build number]\conf used when packaging a message. The default file is **default\_send\_handler.wsdd**. A user-defined WSDD file can be specified for outbound messages. However, doing so requires engaging a professional services consultant, obtaining the optional Software Development Kit or both.

### Payload location

The payload transferred in a SOAP message can be in the SOAP body or packaged as a MIME attachment.

By default outbound payloads are placed in the SOAP body of the packaged message. The payload also can be packaged as an attachment by changing this field value. Setting the payload location to **none** has the effect of not attaching the payload as an attachment or in the SOAP body. In the **none** case, a SOAP handler must be configured to attach the payload where desired.

To use the default configuration, set the payload location to **SOAP body**.

### SOAP action

The SOAP action header value. If specified for outbound messages, the value is used when packaging. If no value is specified, SOAPAction defaults to a blank string.

To use the default configuration, leave this field blank.

Note that in the default configuration, WS-Addressing header is added. Included are the sender and receiver routing IDs and a message ID.

### Encrypt messages

Select this check box to encrypt outbound messages.

To use the default configuration, make sure this check box is selected and do not change values of any of the related fields.

#### *Message encryption algorithm*

Supported encryption algorithms are Triple DES (default) and AES.

***Encrypt attachments***

Sets whether attachments are encrypted (on by default).

***Encrypt SOAP body***

Sets whether the SOAP body is encrypted (on by default).

***XPaths to encrypt***

You can add specific SOAP envelope elements and sub-elements to be encrypted. For each you add, specify the namespace prefix, namespace URI and local XPath.

An XPath example is:

Namespace prefix: soapenv

Namespace URI: <http://schemas.xmlsoap.org/soap/envelope/>

Local XPath: Body

***ID refs to encrypt***

You can add specific SOAP envelope elements and sub-elements to be encrypted.

**Sign messages. Partners use your certificate to verify you as the sender**

Select this check box to digitally sign the messages you send.

To use the default configuration, make sure this check box is selected and do not change values of any of the related fields.

***Sign attachments***

Sets whether attachments are signed (on by default).

***Sign SOAP body***

Sets whether the SOAP body is signed (on by default).

***XPaths to sign***

You can add specific SOAP envelope elements and sub-elements to be signed. For each you add, specify the namespace prefix, namespace URI and local XPath.

XPath examples are:

Namespace prefix: soapenv  
Namespace URI: <http://schemas.xmlsoap.org/soap/envelope/>  
Local XPath: Header

Namespace prefix: soapenv  
Namespace URI: <http://schemas.xmlsoap.org/soap/envelope/>  
Local XPath: Body

***ID refs to sign***

You can add specific SOAP envelope elements and sub-elements to be signed.

## ***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 143 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 [Delete, resend, reprocess options](#) on page 506.

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 295).



**Figure 143. 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 451.

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 144.

**1) Choose the settings to specialize:**

- Pick the sender routing ID
- 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 Generic 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 sending rules for the Web services 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 144. Community collaboration settings options**

To specify how a community sends messages to a specific partner, see [Partner collaboration settings](#) on page 477.

The following describes the override options.

**Pick the sender routing ID**

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 451.

**Set sending rules for the AS2 message protocol**

Lets you override the AS2 global defaults. For field descriptions see [AS2 default settings](#) on page 454.

**Set sending rules for the AS3 message protocol**

Lets you override the AS3 global defaults. For field descriptions see [AS3 default settings](#) on page 459.

**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 463.

### **Set sending rules for the Generic e-mail message protocol**

Lets you override the secure e-mail global defaults. For field descriptions see [Generic e-mail default settings](#) on page 466.

### **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 467.

### **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 468.

### **Set sending rules for the Web services message protocol**

Lets you override the web services global defaults. For field descriptions see [Web services default settings](#) on page 470.

### **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 473.

### **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 144.

**1) Choose the settings to specialize:**

- Pick the sender routing ID
- Pick the receiver routing ID
- 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 Generic 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 sending rules for the Web services message protocol
- 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
- Specify the delivery exchange to send messages to

**Figure 145. Partner collaboration settings options**

The following describes the override options.

**Pick the sender routing ID**

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 receiver routing ID**

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 451.

**Set sending rules for the AS2 message protocol**

Lets you override the AS2 global defaults. For field descriptions see [AS2 default settings](#) on page 454.

**Set sending rules for the AS3 message protocol**

Lets you override the AS2 global defaults. For field descriptions see [AS3 default settings](#) on page 459.

**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 463.

**Set sending rules for the Generic e-mail message protocol**

Lets you override the secure e-mail global defaults. For field descriptions see [Generic e-mail default settings](#) on page 466.

**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 467.

**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 468.

**Set sending rules for the Web services message protocol**

Lets you override the web services send handler config file global default. For field descriptions see [Web services default settings](#) on page 470.

**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 473.

**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.

**Specify the delivery exchange to send messages to**

Lets you choose the transport for sending messages to this partner.



# 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 447.

## Concepts

- [Opening the validation page](#)
- [Duplicate EDI documents](#)
- [Signed or unsigned messages](#) on page 483
- [Encrypted or plain text messages](#) on page 484
- [Duplicate CSOS orders](#) on page 485

## 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.

Duplicate messages    Signing    Encryption

Reject duplicate messages  
 Allow duplicate messages

Add an exception for a partner

Save changes

**Figure 146. 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.

The screenshot shows a user interface for configuring message validation rules. At the top, there are three tabs: 'Duplicate messages' (selected), 'Signing' (highlighted in blue), and 'Encryption'. Below the tabs, there are two radio button options: 'Reject messages that are not signed' (selected) and 'Accept messages that are not signed'. A link 'Add an exception for a partner' is located below the radio buttons. At the bottom of the panel is a 'Save changes' button.

**Figure 147. 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 accept 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.

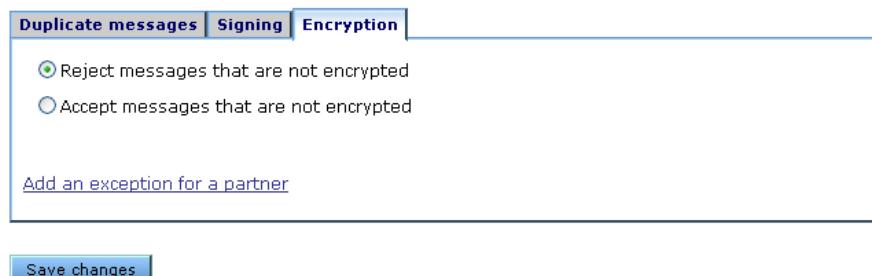
The default selection is to reject messages that are not encrypted. This means the messages a partner sends must be encrypted with your public encryption key. Your community decrypts the message with the private key in the digital certificate associated with your community profile. To send encrypted 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.



**Figure 148. 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](#)

## Procedure

- [Define message attributes](#) on page 491
- [Define message actions](#) on page 492

## 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. <a href="#">Add a definition.</a>	

**Figure 149. 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. (Note that inbound receipts are processed before reaching the message handler. So Is Receipt is applicable as a message action only for outbound receipts.)

## 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:

<b>Sender ID</b>	<b>equals</b>	<b>Community A</b>
(attribute)	(operator)	(value)

You can set up multiple conditions, all of which must be met for the action to trigger.

The following defines operators. Not all of these are available for all attributes.

**Exists** operator tests whether a particular attribute has any value at all.

**Doesn't exist** operator tests whether a particular attribute has no value at all.

**Equals** operator tests whether a particular attribute has a \*specific\* value. (Meaning that it both 'Exists' and has that specific value.)

**Not Equal** operator tests whether a particular attribute does not match a specific value, meaning that it exists but not with that specific value.

# 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 492.

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 487.

## 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 491.

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 487.
- 4 Specify an operator and value for triggering the action and click **Next**. See [Define actions](#) on page 490 for definitions of operators.
- 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 is a tool in the user interface for monitoring database records of traded messages. You can search for and view payloads and receipts. Select **Message tracker** on the top toolbar to open the search page.

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 515.

## Concepts

- [Message Tracker search controls](#)
- [Search results](#) on page 501
- [Delete, resend, reprocess options](#) on page 506
- [Sharing saved searches](#) on page 508
- [Changing search default settings](#) on page 508
- [Message details](#) on page 503
- [Message receipts](#) on page 505
- [Configure payload view](#) on page 511

## Message Tracker search controls

Controls on the **Custom search** panel on the left side of the main Message Tracker page let you search for messages by conditions you specify. Messages matching your search conditions are displayed on the search results area of the page (see [Search results](#) on page 501).

## Message Tracker page

Figure 150 shows a partial view of the main Message Tracker page as it may appear when first opened. Due to length, only the top portion of the custom search panel on the left side of the page is shown.

The screenshot shows the 'Custom search' panel on the left and the 'Search results' table on the right.

**Custom search:**

- Search name:
- New | Save | Remove | Find
- Messages per page: 100
- Maximum # of search results: 1000
- Trading information:**
  - From:  Community or partner name or routing id
  - To:  Community or partner name or routing id
  - Status: Any
  - Direction: Any
  - Consumption URL:
  - Original filename:

**Search results:**

- Showing 0 - 0 of 0 messages
- Select an action...
- Click a table cell for more filtering options
- Table headers:  Type,  Origination, From ID, To ID
- Please define a search.

Figure 150. Partial view of Message Tracker search page

## Custom search

The following describes the controls and fields on the custom search panel on the left side of the main Message Tracker page.

### Search name

If you leave the field blank and click **Find**, messages matching conditions set in the trading information and date areas on the custom search panel are searched for. Once search results are displayed, you can type a name for the search and click **Save**. Later you can run the same search again by selecting **Message tracker > [name of saved search]** under **My searches** on the menu.

### New

**New** is a clear action. Use this button to clear the page of search results and begin a search from scratch.

### Save

**Save** lets you save the conditions for a search you have performed. This button 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** menu list.

### **Find**

**Find** searches for all records matching the conditions specified on the search page, if any have been specified. If no conditions are specified, the default action is to search for all messages traded within the number of days specified in the [Default # of days to search for](#) field on the Message Tracker global settings page (see [Changing search default settings](#) on page 508).

### **Messages per page**

The value of this field is the maximum number of search results to display per search results page. If the number of found messages exceeds the page limit, the results are displayed across multiple pages.

You can set this maximum on a search-by-search basis.

### **Maximum # of search results**

The value of this field is the maximum number of messages that will return after a search is executed. If a search finds more than this number, the results are trimmed to return only the number up to the maximum.

You can set this value on a search-by-search basis, but only to the maximum allowed by a field on the Message Tracker global settings page. You also can change the default value of this field on the global settings page. For more information see [Changing search default settings](#) on page 508.

## ***Trading information***

The following describes the fields under the trading information heading. Use of these fields is optional.

**From**

The message sender. You can type a community or partner name or routing ID. Or, click the ellipsis button to select a party.

**To**

The message receiver. You can type a community or partner name or routing ID. Or, click the ellipsis button to select a party.

**Status**

The status of the messages to search for: any, delivered, failed, ignored, in process, negative response, resubmitted, resubmitted original, scheduled production, split and waiting for receipt.

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).

If you search for negative response, Message Tracker returns payloads with negative responses from partners. And if [Hide receipts](#) is turned off, a search also returns negative response receipts.

If you search for messages with delivered status, negative responses also are returned. This is because a negative response is a delivered state.

**Direction**

The direction of the messages to search for: 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.

If you perform ePedigree processing, a search for **internal** returns ePedigree messages.

A search for **external** does not yield useful results.

**Consumption URL**

Allows you to search for messages based on the transport used by a community to pick up messages from integration or receive messages from partners.

For example, if a community uses file system integration, copy the pickup location (such as C:\data\ediout) from the transport maintenance page and paste the path in the consumption URL field in Message Tracker.

**Original filename**

The original name of the message file received from a partner or picked up from integration. Using this in conjunction with date or time conditions may result in more useful search results.

**Delivery filename**

The name of the message file sent to a partner or delivered to integration. Using this in conjunction with date or time conditions may result in more useful search results.

**Document ID**

The control ID of an EDI document.

**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**

A unique identification the trading engine generates and assigns to a message.

### **Core ID**

A unique identification the trading engine generates and assigns to a message. Core IDs are found in log files.

### **Conversation ID**

Useful when searching for ebXML messages.

### **Integration ID**

If you have a back-end system or an inline process that supplies a specific value to the available IntegrationId meta-data element, you can include the integration ID in a search.

### **Hide receipts**

Show or hide message receipts in search results.

### **Hide subordinate messages**

Show or hide subordinate messages in search results. A subordinate message is one that was split from a parent document. A subordinate message also can be described as a child message of a parent document.

In ebXML trading if hide subordinates is turned off, Message Tracker reports two payload messages: the original MMD picked up from integration and the message split from the original MMD. If hide subordinates is turned on, only the original MMD picked up from integration is reported.

### **Hide pings**

Show or hide peer network ping messages in search results. This option applies only if you are licensed to use the peer network.

## Date

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.

The default settings for searching by date is the **Origination** date **Within the last [n] days**. If you open the Message Tracker search page and perform a search without changing the date defaults, you may not get the results you expect if many database records are older than the number of days specified in the **Within the last [n] days** option.

You can change the default value of the option **Within the last [n] days** on the Message Tracker global settings page (see [Changing search default settings](#) on page 508).

## Columns

To adjust the column display of the search results, expand the **Columns** area on the left side below the **Date** area and select the columns you want to show or hide on search results pages. Any custom column display you set up applies only to you and not to other users.

Under the columns check boxes you can select the order for sorting search results.

# Search results

Once search results are displayed, you can enlarge the results display area by clicking the **Show/Hide** tab to the right of the search name field.

Figure 151 is an example of a search results page with the controls hidden. Click **Show/Hide** to toggle the view.

**Search results:**  
Results as of: Monday, January 8, 2007 9:55:53 AM MST  
Restricted to messages originating between: 1/1/07 9:55 AM and 1/8/07 9:55 AM  
Showing 1 - 10 of 24 messages

Result page: 1 2 3 [Next](#)

<input type="checkbox"/>	Type	Origination	From ID	To ID	Status
<a href="#">Details</a>	Payload	Jan 4, 2007 03:58:33 PM MST	ZZworldwide	ZZacme	Delivered
<a href="#">Details</a>	Payload	Jan 4, 2007 03:58:33 PM MST	ZZworldwide	ZZacme	Delivered
<a href="#">Details</a>	Payload	Jan 4, 2007 03:58:33 PM MST	ZZworldwide	ZZacme	Delivered
<a href="#">Details</a>	Payload	Jan 4, 2007 03:58:33 PM MST	ZZworldwide	ZZacme	Delivered
<a href="#">Details</a>	Payload	Jan 4, 2007 03:58:33 PM MST	ZZworldwide	ZZacme	Delivered
<a href="#">Details</a>	Payload	Jan 4, 2007 03:58:33 PM MST	ZZworldwide	ZZacme	Delivered
<a href="#">Details</a>	Payload	Jan 4, 2007 03:58:33 PM MST	ZZworldwide	ZZacme	Delivered
<a href="#">Details</a>	Payload	Jan 4, 2007 03:58:33 PM MST	ZZworldwide	ZZacme	Delivered
<a href="#">Details</a>	Payload	Jan 4, 2007 03:58:32 PM MST	ZZworldwide	ZZacme	Delivered
<a href="#">Details</a>	Payload	Jan 4, 2007 03:58:32 PM MST	ZZworldwide	ZZacme	Delivered

Result page: 1 2 3 [Next](#)

**Figure 151. Search results page in Message Tracker**

You can click some of the values in the returned search results to display options for searches based on a single value. Move your cursor over the table of search results. When the cursor displays as a hand symbol, you can click the table cell for more searching options.

For example, in Figure 152 the **From ID** value **ZZacme** was clicked to display two search options.

<input type="checkbox"/>	Type	Origination	From ID	To ID	Status
<a href="#">Details</a>	Receipt	Jan 4, 2007 03:58:37 PM MST	ZZacme	ZZworldwide	Delivered
<a href="#">Details</a>	Receipt	Jan 4, 2007 03:58:37 PM MST	ZZa		Delivered
<a href="#">Details</a>	Receipt	Jan 4, 2007 03:58:36 PM MST	ZZacme	ZZworldwide	Delivered

**Figure 152. Click a table cell to display more search options**

These search options are:

### Add this item to the filter

Select to add this value as a search condition in the trading information area of the custom search panel and perform a new search. You can select this to cumulatively add values for search different conditions. The value you select is added to the search conditions already set up on the custom search panel.

### Filter by only this item

Select to clear any search conditions in the trading information area of the custom search panel and perform a new search based only on this value. You can select this to clear all search conditions except one. This also resets the date condition to the default of **Origination** and **Within the last [n] days**.

After executing a search, 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. However, you can share your searches with other users (see [Sharing saved searches](#) on page 508).

## Message details

Once search results are displayed, you can view message details. Click **Details** to open the message details page for the selected payload or receipt. Trading activity information is provided on the following tabs.

### Document summary

This tab identifies the message type and status (delivered, negative response, failed, in process, waiting for receipt, resubmitted). It also reports the message direction (for example, inbound from a partner or outbound to a partner). If a failed message, a reason is given at the bottom of the tab.

Links are provided for viewing and downloading the message payload.

You can click **Add a note** to type notes regarding the message.

## Message processing details

This tab graphically shows the processing steps for the message. It also provides details about signing, encryption, compression and whether there was a receipt for the message. If you back up copies of messages, the name of the backed up file is given.

### ebXML activity

This tab reports meta-data specific to ebXML messages. The information is view-only. Use the [Message attributes](#) tab to perform meta-data searches.

This tab only displays for an ebXML message. If you do not process ebXML messages, this tab does not display.

### RosettaNet activity

This tab reports meta-data specific to RosettaNet messages. The information is view-only. Use the [Message attributes](#) tab to perform meta-data searches.

This tab only displays for an RosettaNet message. If you do not process RosettaNet messages, this tab does not display.

### Document activity

This tab provides the dates and times for events related to a message's processing history.

### Message attributes

This tab reports values for meta-data attributes related to messages. You can use the meta-data to perform narrower message searches based on specific values.

Some of the meta-data on this tab is self-explanatory (for example, Direction, ReceiverRoutingId, SenderRoutingId). Other meta-data may seem esoteric. Also, the list of data can vary depending on the message type. You may find this tab provides more information than needed for monitoring normal message traffic. But if you are working with technical support to troubleshoot an issue, support may refer you to this tab for information. This meta-data also may be helpful for users who employ APIs to manipulate message meta-data or who exchange messages via the ebXML message protocol.

The table on this tab shows the meta-data names or the attribute names. A meta-data name is the name the trading engine uses internally. An attribute name is a friendly name for a meta-data element. For example, **ConsumptionUrl** is the meta-data name and **Consumption URL** is the attribute name. You can toggle between name types by clicking **Show metadata names** and **Show attribute names**.

You can select meta-data to include in a new search, and you can include the meta-data element in saved searches. If you clear the check box for **Restrict search to messages that originated within the last [n] days**, the new search will include all dates. The number of days for this option can be changed (see [Changing search default settings](#) on page 508).

## Icons on search results page

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.

## 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 447).

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.

## Delete, resend, reprocess options

The search results page has three action options that can be applied to the displayed records. These are: delete, resend and reprocess.

Not all actions are valid for all types of messages, but the user interface indicates when an action you select can be applied. If an action is valid for a message, you can select the check box at the beginning of the results record line. If an action is not applicable, you cannot select the check box.

### Delete

Delete messages and receipts from the database and backup directory. Only messages in a final processing state can be deleted. Only parent messages can be deleted. To delete child messages or receipts, the related parent must be selected.

### Resend

The following can be resent:

**Receipts for messages received previously from partners** can be resent to partners. Resent receipts are not packaged (no signing, encryption, compression).

**Messages previously sent to partners** can be resent to partners. A previously sent message is not repackaged. Rather, the previously sent packaged message is sent again. Transport retries and business protocol resends are respected for the resent message, as if the original send did not occur.

**Messages previously sent to integration** can be resent to integration. Transport retries are respected for the resent message.

### Reprocess

Reprocessing is applicable to messages outbound to partners and inbound from partners.

**Reprocessing an inbound message** again unpacks the received message, resends the message to integration and resends the receipt to the partner if the receipt is not synchronous.

**Reprocessing an outbound message** repackages the original message and resends it to the partner.

If you resend a message a partner already has acknowledged, the partner's trading engine may reject the resent message as a duplicate unless steps are taken to anticipate the resend.

When a message is resent or reprocessed, 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.

For reprocessing, if an original outbound 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.

**Note:** Whether you select messages to resend or reprocess, Message Tracker reports the status of both actions as **resubmitted**. For the purpose of reporting status, Message Tracker cannot differentiate between a resent message and a reprocessed message. Instead, the status of both actions is given as resubmitted.

## Sharing saved searches

The **Share my searches** menu option opens a page that lets you share your saved searches with other users. This allows other users to execute searches without having to recreate the conditions you already have set up for saved searches.

Select the saved searches you want to share and click **Share**.

If another user has a saved search with a name identical to one of your saved searches, the other user's saved search name is listed under the **Searches with names duplicating my searches** column. This only means the names are the same; the search conditions could be quite different. Select the overwrite check box if you want the saved search you share to replace a saved search of the same name belonging to another user.

## Changing search default settings

Use the Message Tracker global settings page to change:

- The list of system-defined default searches that appear on the Message Tracker menu.
- Values that control the maximum number of search results that can be displayed or deleted.

To open the page, select **System management** on the top toolbar to open the System management page. Click the link named **Configure global message tracker settings** to open the default settings page.

## Global message tracker settings

### Default menu items

- All messages
- All messages-Last 1 Hour
- All messages-Last 2 Hours
- All messages-Last 6 Hours
- All messages-Last 24 Hours
- Failed messages
- Negative Response

### Query restrictions

Default # of days to search for:	<input type="text" value="7"/>
Maximum # of messages to purge:	<input type="text" value="1000"/>
Default # of search results to return:	<input type="text" value="1000"/>
Maximum # of search results to return:	<input type="text" value="20000"/>

**Save changes**

**Figure 153. Page for setting search defaults**

The following describes the fields on the Global message tracker settings page. Click **Save changes** after making any changes.

## Default menu items

The following check boxes control whether system-defined default searches are available to select under **Message Tracker > Message searches**. If selected, these searches are available for all users. Although you can choose whether to make these available universally, you cannot change the search conditions of system-defined default searches.

### All messages

This default search finds all messages. If the database has a large volume of records, you may want to turn off this search.

### All messages-Last 1 Hour

This default search finds all messages traded in the last hour.

### All messages-Last 2 Hours

This default search finds all messages traded in the last two hours.

### All messages-Last 6 Hours

This default search finds all messages traded in the last six hours.

### All messages-Last 24 Hours

This default search finds all messages traded in the last 24 hours.

### Failed messages

This default search finds all messages with a status of failed.

### Negative Response

This default search finds all messages that received a negative response from a partner.

## Query restrictions

### Default # of days to search for

The value in this field is the number of days that appears in the search option **Within the last [n] days** under the date area on the custom search panel of the Message Tracker page.

This also is the value for a search condition on the message attributes tab of the message details page. The value applies to the check box for **Restrict search to messages that originated within the last [n] days** at the bottom of the tab.

Moreover, if you click **Find** without specifying search conditions on the Message Tracker search page, the search finds all messages traded within this number of days.

### Maximum # of messages to purge

The maximum number of search results you can delete with a single delete action on the Message Tracker page. For example, if more than the maximum number of search results are returned and you try to delete all results, the trading engine will not delete any of the messages. Instead you will be prompted to define a search that returns fewer records than the maximum to purge.

Limiting the number of search results that can be deleted at once helps maintain database performance.

### Default # of search results to return

The value in this field is the default value of the **Maximum # of search results** field on the custom search panel of the Message Tracker page.

### Maximum # of search results to return

The value in this field is the highest allowed value for **Maximum # of search results** field on the custom search panel of the Message Tracker page. If you are performing a search and enter a number in the Maximum # of search results field larger than this value, an error message displays.

## 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 513 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.	
<input checked="" type="checkbox"/> <a href="#">Add a document type</a>	

**Figure 154. 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:<sup>\*</sup>

Description:

Stylesheet:<sup>\*</sup>

Choose how you want to view this document type by default

View as text  
 View formatted using a stylesheet

**Figure 155. 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 backed-up 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 295 for more information.

**Note:** If licensed to use Synchrony ePedigree, you cannot configure the trading engine to delete pedigree database records. Pedigree records are not affected by using the page in the user interface for deleting messages or by the messagePurgeTool utility.

## Concepts

- [Global backup configuration](#)
- [Delete unwanted files and records](#) on page 518
- [Purge all records, backup files](#) on page 520
- [Configure event purging](#) on page 521

# Global backup configuration

A message has the potential to be backed up multiple times: When first consumed and thereafter when the content and state have been changed. Integration pick up exchanges and exchanges for receiving messages from partners are regarded as consumption exchanges. There are a number of opportunities for backing up a message following a content and state change. For example, both occur after a message received from a partner has been unpackaged.

There are two places in the user interface to configure how the trading engine backs up the messages it processes.

- 1** Message backups can be turned on or off selectively for each integration delivery exchange that picks up from, or delivers to, a back-end system. You also can turn backing up on or off for exchanges that receive messages over the Internet from partners. Backing up is on by default. To change the backup behavior for an exchange, see [Transport maintenance](#) on page 295.
- 2** The global backup configuration page has settings affecting how all messages for all communities are backed up.

This topic describes using the global backup configuration page. But to meet your needs, you may have to adjust fields on this page as well as tune backup controls at the transport level.

## Field descriptions

To open the global backup configuration page, click **Trading configuration** on the toolbar. On the pick a community page, click **Configure the global backup settings**. The following describes the fields.

### Root path

Use the root path field to specify the path of the backup directory. The backup directory can store copies of all traded documents in their clear (unencrypted) and encrypted forms.

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

This 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.

### Message handler backs up messages

This check box specifies whether to back up messages that pass through the message handler within the trading engine. The message handler is responsible for actions such as packaging (encryption, adding MIME headers) and unpackaging.

Turning on message handler backups (the default) does not result in adding a larger volume of files to the backup directory, as messages are backed up when content and state have changed, not simply when the state has changed. However, turning off backups can result in fewer backed-up states. Regardless, turning this on or off does not affect searching in Message Tracker.

If security of clear text files is an issue, turn off message handler backups. This ensures all messages — encrypted and unencrypted — passing through the handler are not backed up. Furthermore, you must disable backups for transports that handle clear text messages (for example, integration pickup). Only disabling both guards against backing up clear text messages.

### States of backed up files

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 labels noting the state of backed-up messages. There are many possible states. Some common states are:

### Consumed

Consumed is appended to names of documents that have been received from partners but not unpackaged or that have been picked up from integration but have not been packaged or sent.

### UnpackagedRequest

UnpackagedRequest is appended to names of documents that have been received from partners and unpackaged. These documents are clear 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 purge dates for trading engine messages**. This displays the page for configuring purge dates (Figure 156).

### Configure purge dates for trading engine messages

Select an interval for assigning dates for purging message records in the database and documents in the back-up directory. The system uses this interval to calculate how many years, months or days from today's date to set the purge dates. The system purges messages on their purge dates, which are assigned the moment messages enter the system.

For example, if the interval is 45 days, the purge date is set as 45 days from today's date for all messages processed today. If you change the interval, future messages entering the system are affected, but the purge dates of messages processed earlier are not reset. You can view the purge date for a message on the details page in Message Tracker.

Purged database records and backed-up documents are deleted permanently. Purged messages cannot be recovered.

Delete messages on their purge dates  
 Turn off purging, but continue setting purge dates

Purge interval:  years  months  days

**Save changes**

**Figure 156. 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 to delete records and documents that have reached their purge dates.

If you change the purge interval, the new interval affects only messages processed after the change. The purge dates of messages processed before the change remain the same. But you can use a tool to change purge dates retroactively in line with the new purge interval. See [Purge all records, backup files](#) on page 520.

Only database records and messages in a final state are deleted. A final state is when no more processing action is pending. Final states are reported in [Message Tracker](#) as: Delivered, Failed and Resubmitted. When a message is resubmitted, the message is marked as resubmitted and a child of the message is spawned. A message with a resubmitted state is deleted only if its child messages also have a final state.

You can choose to turn off purging of database records and 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))
```

We recommend setting the age for deleting messages identical to the age for deleting message-related events. For more information, see [Configure event purging](#) on page 521.

## Note for CSOS users

If you are licensed for CSOS functionality, CSOS records and message backups are scheduled to be retained for two years, beginning with version 5.4. The CSOS retention period is configurable, but only with the help of Axway.

If the page for configuring purge dates calls for deleting records and messages less than 2 years old, CSOS records and messages will not abide by this schedule and will be retained for two years. If the setting on this page is more than two years, CSOS messages will be retained for the longer period.

# 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.

---

The messagePurgeTool can be run with one of the following parameters. You can only use one parameter at a time.

## **deleteAll**

This deletes all records in the database and related backup files regardless of age or state.

Depending on the volume of records in the database and backup files, the utility may have to run a while to delete all.

## **resetMessages**

This is for changing the purge dates of records in the database and documents in the back-up directory. When messages are processed, the trading engine assigns future purge dates, based on the age interval set on the purge configuration page in the user interface. If you change the age interval, you can use the resetMessages option to change purge dates of existing records and documents in line with the changed interval.

For example, if the interval is 45 days, the system sets the purge date for messages processed today as the date 45 days in the future. On day 44, you change the interval to 90 days and then run the resetMessages option. The system re-calculates the purge dates of existing messages as 90 days in the future of the messages' origination dates. This means on day 44, the purge date will be set ahead 46 additional days, for a total of 90 days before purging.

The utility may take a while to run if there are a large number of records to reset.

See [Delete unwanted files and records](#) on page 518.

For example, to delete all records in the database and all related backup files, run the following command:

**messagePurgeTool -deleteAll**

Events related to running the messagePurgeTool are written to the messagePurgeTool.log file in the logs directory.

# Configure event purging

You can configure the trading engine to delete unwanted database records for message-related events based on the age of the records.

Go to the trading configuration area of the user interface and click **Configure purge dates for events**. This displays the page for configuring purge dates (Figure 157).

## Configure purge dates for events

Select an interval for purging events from the database. The system uses this interval to calculate how many years, months and days from today's date an event should be purged.

For example, if the interval is 45 days, all events 45 days and older from the time they were generated will be deleted.

Purged database records are deleted permanently. Purged events cannot be recovered.

We recommend setting the age for deleting message-related events identical to the age for deleting trading engine messages.

Enable purging events

Disable purging events

Purge interval:  years  months  days

**Save changes**

**Figure 157. Settings for deleting message-related events**

The default configuration is to delete database records for message-related events after 45 days. You can use the years, months and days fields to change the age settings. The system checks every 15 minutes to delete events that have reached the age threshold.

We recommend setting the age for deleting message-related events identical to the age for deleting trading engine messages. For more information, see [Purge all records, backup files](#) on page 520.





# 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

- ◆ [Levels of scripting](#)
- ◆ [Editing the command set document](#) on page 524
- ◆ [FTP tester tool](#) on page 525

## 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. More information is available in the Developer's Guide for the optional Software Development Kit available from Axway.
- 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. More information is available in the Developer's Guide for the optional Software Development Kit available from Axway.

## 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.

## 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, DELETED, LLIST, ASCII,
BIN, LCD, QUIT): ?
CONSUMER metacommands (abbreviations shown in upper case):
?                               help
LIST                          list files on host
RECeive filename   retrieve file from host
DELETED 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, DELETED, 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 or with Axway using the Quickstart partner profile.

Procedures are provided for configuring the trading engine to exchange test documents with a trading partner or Axway. Testing with Axway is recommended for customers who have purchased support to validate your organization's implementation of the system.

Available at all times, our test server provides you with a correctly configured trading partner. By using it you can validate your community profile and communications against a thoroughly tested system.

## Procedure

- ◆ [Configure for test trading](#)
- ◆ [Start the test](#) on page 531
- ◆ [Troubleshooting for e-mail testing](#) on page 532
- ◆ [Complete the testing](#) on page 533

## 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 160.

If you want to test trade with Axway, you can configure the community profile to receive messages using any of the supported delivery exchanges. If you need guidance on which transport method to use, contact technical support. For information about transport options see [Delivery exchanges](#) on page 201.

If you want to test with a trading partner, 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 223.

- 2** Generate a test self-signed certificate for your test community profile. See [Set up certificates for a community](#) on page 407.
- 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 want to test with Axway, send the e-mail to the address Axway provides you. Type **Request for Quickstart Test** in the subject line of your e-mail.

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.

If you want to test with Axway, import the Quickstart partner profile file. This file is at [install directory]\[build number]\profiles\staged\partner.

If you want to test with a trading partner, have the partner send the profile file to you by e-mail or another method. Then import the profile. If the partner also uses 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 164  
[Import a partner profile](#) on page 168.  
[Import certificates for partners](#) on page 421

- 5 In the partner profile, select a default transport for sending test documents. If there is only one transport, that is the default.

If you imported the Quickstart profile, select the delivery exchange you want to use for sending messages to Axway. On the Quickstart partner summary page, click **Delivery exchange** on the navigation graphic to display a page that lists the available transports for sending. 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.

If testing with a partner and there is more than one transport for sending in the partner profile, you and your partner should decide which one to use. Then select the appropriate transport to use as the default for sending in the partner profile.

### Pick a delivery exchange

These are the delivery exchanges for sending messages to this partner. If there are multiple exchanges, the one at the top of the list is the default exchange. The system only uses the default exchange. Use the up and down arrows to change the default exchange.

**Partner:** *Quickstart*

Type	Location	Up	Down	Remove
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 158. Example of available transports for sending in partner profile**

- 6 Contact Axway or 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 535 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 495.
- 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 Axway or 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.

Also, delete the Quickstart partner profile if you tested with Axway.





# 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 535
- [Run from a command line](#) on page 538

## 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.

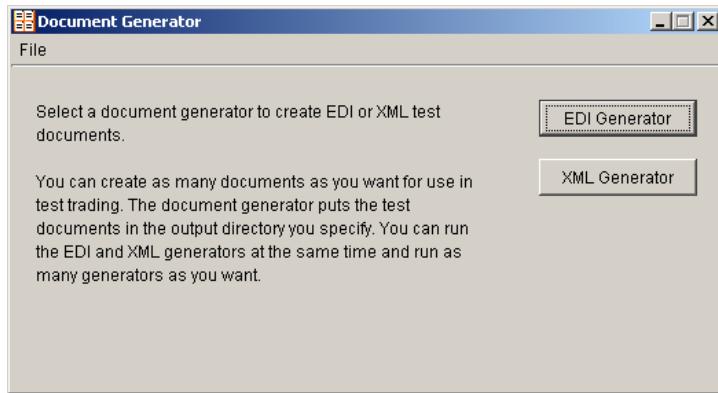
## Steps

- 1 In Windows select **Start > Programs > Axway > Document Generator.**

In UNIX log in to the axway 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:

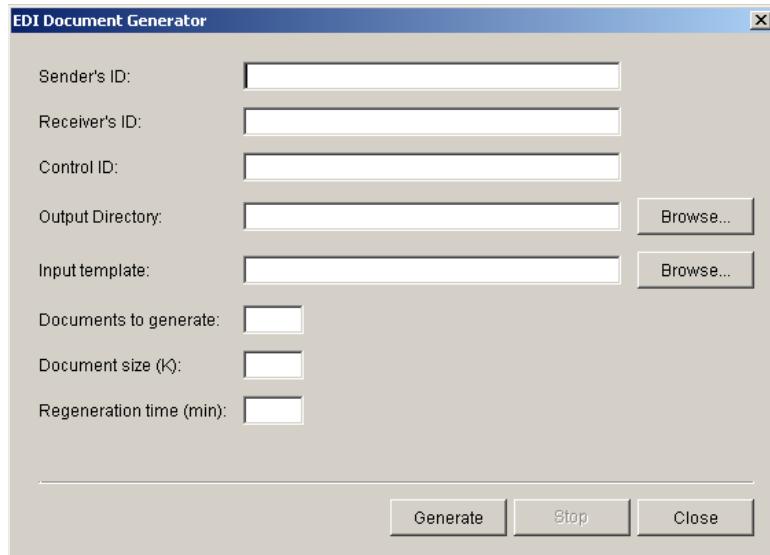
**[install directory]/[build number]/bin/docGen**

You also can run the Document Generator from a command line. See [Run from a command line](#) on page 538.



**Figure 159. 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 160. EDI Document Generator window**

- 3 Complete the fields. See [Field descriptions](#) on page 536.
- 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 535.

**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 535.

## 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 <b>EDI</b> or <b>XML</b> .	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 <b>1</b> and <b>999999</b> to indicate the size of each document you want to create.	Required

Parameter	Description	Usage
-ndocs	Any value between <b>1</b> and <b>999999</b> 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 <b>1</b> and <b>999999</b> 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/axway/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](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](http://www.ebxml.org) or [www.oasis-open.org](http://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

- ◆ [ebXML message lifecycle](#) on page 542
- ◆ [ebXML message meta-data](#) on page 544
- ◆ [Message meta-data documents](#) on page 551
- ◆ [Supported trading transports](#) on page 554
- ◆ [Supported integration transports](#) on page 555
- ◆ [Setting up a community for ebXML trading](#) on page 559
- ◆ [Sending ebXML via an intermediary](#) on page 560
- ◆ [Managing CPAs](#) on page 565
- ◆ [Tools for CPAs](#) on page 570
- ◆ [ebXML troubleshooting](#) on page 582

## Procedure

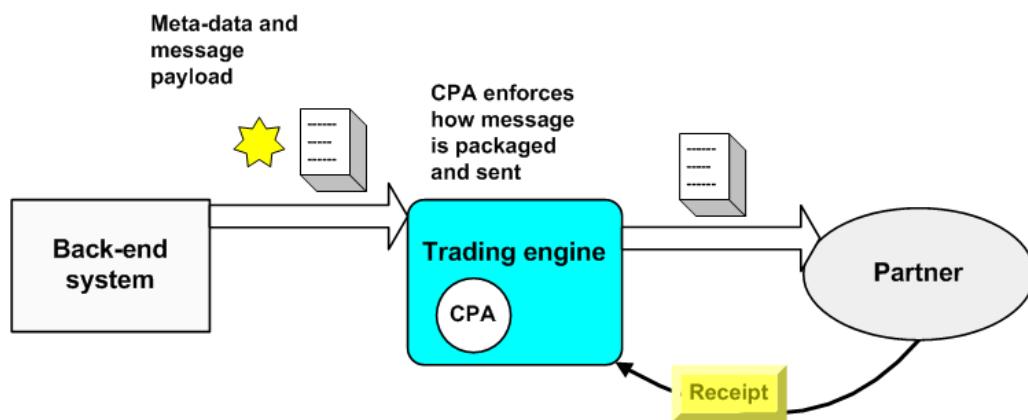
- ◆ [STAR BODs with ebXML](#) on page 574
- ◆ [HL7 payloads with ebXML](#) on page 576
- ◆ [Self-registration of ebXML partners](#) on page 577

# 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 161). There are variations to this description, but the following is a typical case.



**Figure 161. 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 551.

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.

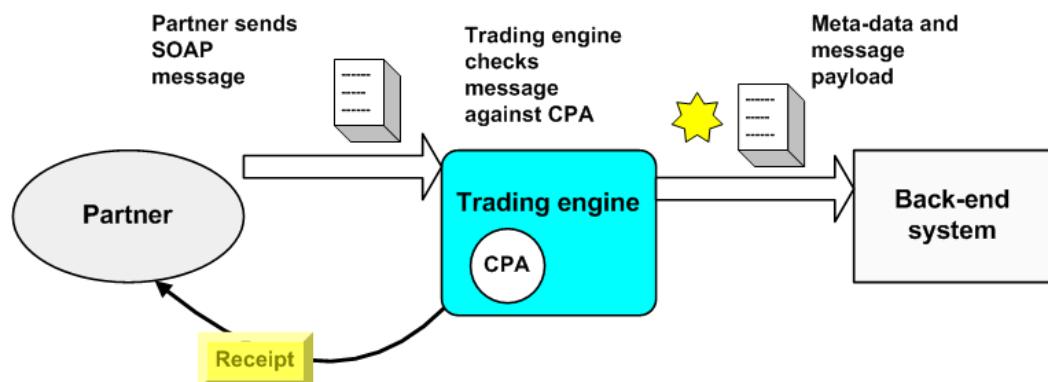
- 4 The trading engine collaboration manager uses the CPA the meta-data identifies to build the binary collaboration rules for packaging and sending the ebXML message to a partner. Packaging usually involves encrypting and signing the message. The collaboration settings in the trading engine user interface 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 162). There are variations to this description, but the following is a typical case.



**Figure 162. 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 555.

## 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 555 explains the role of these transports in more detail.

## ***ebXML meta-data descriptions***

The following describes the meta-data elements for exXML.

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.

### **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 574.

**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.

**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.

### **RemovePayloadAfterProcessing**

Indicates (**true** or **false**) whether the trading engine deletes the payload from the file system after processing the message. (MMDs always are deleted after processing.)

Use of this element is optional. If not used, the payload is not deleted, which is the same behavior as using the element with a value of **false**. If RemovePayloadAfterProcessing is **true**, payloads are deleted after being picked up.

This also works for the resubmit case in which payloads are retrieved from the backup directory.

### **PayloadLocation**

The directory path to the document.

### **PayloadLocationType**

The path and file name of the payload. Payloads can be on a file system or an HTTP or FTP server, as the following examples illustrate:

File system

```
<Location type="filePath">C:\smallxmlPO.xml</Location>
```

HTTP

```
<Location type="xs:anyURI">http://www.oasis-open.org/committees/ebxml-cppa/schema/cpp-cpa-2_0.xsd</Location>
```

FTP

```
<Location type="xs:anyURI">ftp://acme:acme@cfoster-dell.cyclonecommerce.com:4021/mailbox/foo.dat</Location>
```

In the FTP example, acme:acme in the URL represents the user name and password for connecting to the FTP server. Note: using an e-mail address as the password is not supported.

### **PayloadMimeTypeId**

An identifier of the payload content.

### **PayloadMimeType**

The MIME type of the payload. 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 545.

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
PayloadMimeTypeId	MMD only
PayloadMimeTypeType	MMD only
ProcessSpecName	Optional
ProcessSpecUUID	Optional
RefToMessageId	Optional
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 544.

```
- <MessageMetadataDocument documentId="Test_B2" protocol="ebXML" protocolVersion="2.0">
  <Metadata name="From" type="string">remerry-ebxml</Metadata>
  <Metadata name="FromRole" type="http://www.starstandard.org/processes/3A4.xml#Initiator">interop</Metadata>
  <Metadata name="To" type="string">esx6-ebxml</Metadata>
  <Metadata name="ToRole" type="http://www.starstandard.org/processes/3A4.xml#Responder">interop</Metadata>
  <Metadata name="Service" type="string">FileTransfer</Metadata>
  <Metadata name="Action" type="string">B1</Metadata>
  <Metadata name="CPAId">remerry-esx6</Metadata>
- <MessagePayloads>
  - <Payload id="0784247">
    <RemovePayloadAfterProcessing>true</RemovePayloadAfterProcessing>
    <MimeType>smallXmlPo</MimeType>
    <MimeType>application/xml</MimeType>
    - <Location type="filePath">
      /Users/remerry/Source/MiscHabooBStuff/ebXMLInterop4Q2004/CpasAndMmds/smallxmlPO_delete.xml
    </Location>
  </Payload>
</MessagePayloads>
</MessageMetadataDocument>
```

**Figure 163. Example of ebXML MMD for outbound message**

## **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="ID6572235551131040785875coronation"
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>M113104078586.792@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="ID268087701131040580929gnaraloo"
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 164 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.

Message ID

```
- <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 164. 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 [Delivery exchange wizard](#) (see [Delivery exchange wizard](#) on page 212). 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 388.

For more information about trading transports, see [Trading delivery exchanges](#) on page 203 and [Adding transports](#) on page 211.

## 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.

Normally a meta-data element accompanying a picked-up payload provides the CPA ID (literally CPAId) of the CPA to use. But the trading engine can dynamically determine a CPA ID based on other meta-data. If, for example, the MMD for an outgoing ebXML message does not include the CPA ID, the following meta-data are used to find the correct CPA: Service, Action, FromRole, ToRole, From and To. If two or more CPAs match these conditions, the trading engine chooses the CPA with the most recent Start date. However, if the MMD contains a ConversationId, the trading engine selects the CPA used for an earlier related document. This process is valid for any integration protocol.

The following topics describe the pickup and delivery integration options.

For more information about integration transports, see [Integration exchanges](#) on page 206 and [Adding transports](#) on page 211.

### ***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 550.

### 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 551.

### JMS

The meta-data must be set as JMS properties. The meta-data are attached to the BytesMessage containing the payload.

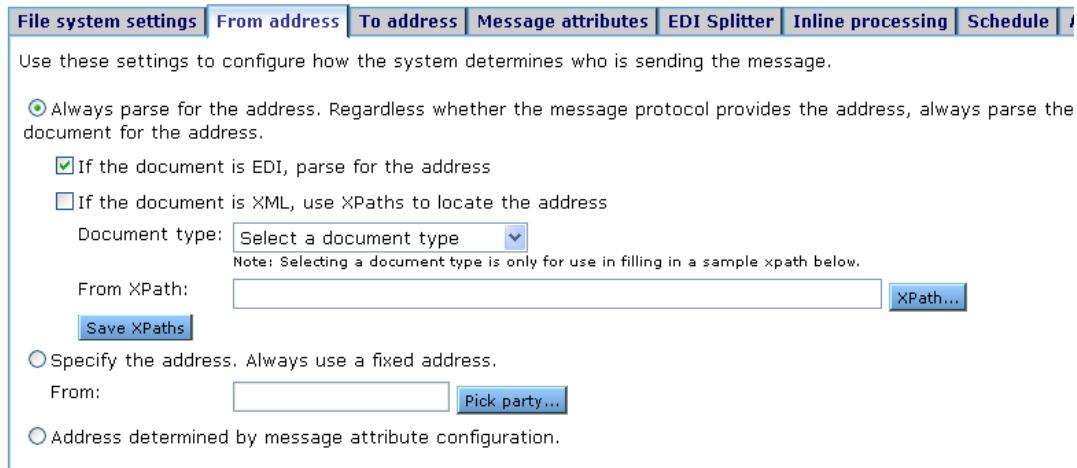
### Web services API server

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 integration](#) on page 255.

## 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.



**Figure 165. From address tab for integration pickup exchange**

Figure 165 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 368

For a list of the minimum name-value pairs, see [Required, optional meta-data for integration](#) on page 550.

## ***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 551.

### **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 166 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 166. 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 156 and [Add a community](#) on page 160 for details not specific to ebXML about setting up 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 577.

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.

## **Sending ebXML via an intermediary**

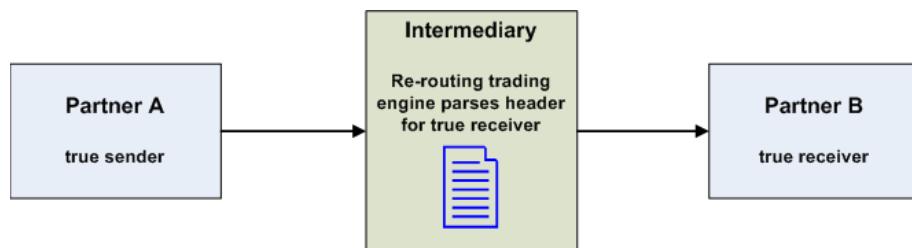
The trading engine has an optional message protocol for sending ebXML messages from a true sender to a true receiver via a third party. The intermediary's role is to receive the message from the true sender and forward it to the true receiver. The intermediary does not use a CPA. However, if both the true sender and true receiver use Interchange or Activator, the end points must use CPAs, as though the message was sent directly.

## Overview of intermediary re-routing

Re-routing is performed by configuring the trading engine in the intermediary role to receive and send messages via the ebXML intermediary message protocol. The protocol is available only if your user license enables it. If the protocol is not listed in the delivery exchange wizard, your license does not support it. For information about the wizard, see [Delivery exchange wizard](#) on page 212.

The CPA used by the true sender must include the URI for connecting to the intermediary instead of the true receiver. Other than this one change, the CPA is written as though the true sender and true receiver exchange messages directly between each other. An exception for the CPA, however, applies when trading is via HTTPS (see [Intermediary trading with HTTPS](#) on page 564).

Figure 167 illustrates the flow for sending an ebXML message via an intermediary. Partner A's trading engine packages and sends the ebXML message in the usual way. Upon receiving the message, the intermediary parses the message header to identify the true receiver, but does not unpack the message. The intermediary then sends the message as received to Partner B. Partner B receives the message as though it was sent directly by Partner A.



**Figure 167. ebXML message sent via an intermediary**

The true sender packages the message as if it is being sent to the true receiver. Except in the case of trading via HTTPS, the true sender's only knowledge of the intermediary is the URI used to send to the intermediary.

After receiving the message from the true sender, the intermediary parses the following meta-data from the packaged ebXML message header:

- ◆ Action
- ◆ ConversationId
- ◆ CPAID
- ◆ ebXML.SignalType
- ◆ ebXML.StatusResponse.RefToMessageId
- ◆ FromRole
- ◆ MessageId

- ◆ ReceiverRoutingId
- ◆ ReceiverRoutingIdType
- ◆ RefToMessageId
- ◆ SenderRoutingId
- ◆ SenderRoutingIdType
- ◆ Service
- ◆ ServiceType
- ◆ ToRole

When the intermediary parses the received message, it only reviews the ebXML header before performing the re-routing. The intermediary re-routes messages internally; it does not write messages to integration. Of the preceding meta-data items parsed from the ebXML message header, the intermediary only needs the sender and receiver IDs and the routing ID types. The intermediary does not use the other meta-data.

The ebXML intermediary message protocol is proprietary to Activator. The protocol does not implement any standards relating to re-routing SOAP or ebXML messages. Activator must be used in the intermediary role, but the true sender and true receiver can use any interoperable trading engine that supports ebXML.

## ***Intermediary message-handling notes***

The following information relates to how the intermediary re-routes ebXML messages:

- ◆ The intermediary does not unpack or repackage messages. The intermediary message protocol supports end-to-end signing and encryption.
- ◆ The intermediary does not validate signatures.
- ◆ Because the intermediary message protocol is exclusively for re-routing ebXML messages, the protocol overrides the message re-routing control a community optionally may have available for determining whether to re-route partners' messages.
- ◆ Receipts are not matched up to the payloads in the intermediary's Message Tracker. The intermediary only passes messages through. No context is kept from message to message.
- ◆ Synchronous acknowledgments, synchronous signals and synchronous business responses are not supported. A SOAP fault is returned to the sending system for any message that requests a synchronous response of any type.

- The intermediary does not check for duplicate messages. The true receiver should perform duplicate checking.
- SOAP faults are not passed from the true receiver back to the true sender. This is because synchronous responses are not supported through the intermediary. Also, there is no standard way to match asynchronous SOAP faults to the original message.

## ***Supported transports***

See [Trading delivery exchanges](#) on page 203 for the transports supported by the ebXML intermediary message protocol.

## ***Configuration of intermediary***

The intermediary does not use CPAs. The intermediary must:

- Configure a delivery exchange in the community profile to receive messages via the ebXML intermediary message protocol from the true sender. The community must have its own unique routing ID. This routing ID is used for logging and tracking messages, but is not included in packaged messages.
- Add a partner profile for the true sender. Use the true sender's routing ID as the profile's routing ID. Add the ebXML intermediary message protocol to the profile.
- Add a partner profile for the true receiver. Use the true receiver's routing ID as the profile's routing ID. Add the ebXML intermediary message protocol to the profile.

Certificates do not have to be associated with the partner profiles for the true sender and true receiver. However, if HTTPS with client authentication is used, each partner's client certificate must be trusted for SSL by the intermediary community. See [Intermediary trading with HTTPS](#) on page 564.

A community profile can be configured to receive messages via both the regular ebXML message protocol and the ebXML intermediary message protocol.

However, a single partner profile can use only one or the other protocol, but not both. Although the user interface does allow for two message protocols to be configured in a single partner profile, the trading engine always uses the first listed protocol to send to the partner. To support

sending point-to-point and re-routing, two partner profiles would be required: One configured using the ebXML intermediary protocol, the other configured using the regular ebXML protocol.

A community acting as an intermediary optionally can perform in-line processing. But such processing must not change the content of re-routed messages. Any change to the content could corrupt the ebXML message.

## ***Configuration of end points***

True senders and true receivers who use Interchange or Activator must use CPAs as though the parties are directly exchanging messages without an intermediary.

Certificates in the CPA must be the certificates of the end points, not the intermediary. The CPA should ignore the intermediary, except the transport end point URIs must point to the ebXML intermediary exchange point configured in the intermediary instance.

## ***Intermediary trading with HTTPS***

If messages are sent via HTTPS, the partners' HTTPS server certificates must be trusted by the intermediary community.

Two end points trading via ebXML through an intermediary can use one CPA. Not only must the TransportReceiver element reference the intermediary's HTTPS URL, the ServerCertificateRef and ClientSecurityDetailsRef elements must point at the intermediary's SSL certificate. The following CPA snippet illustrates these points.

```
<tp:Transport tp:transportId="transportHttps">
  <tp:TransportSender>
    <!-- References endpoint certificate -->
    <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
    <tp:TransportClientSecurity>
      <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:Transport
          SecurityProtocol>
      <tp:ClientCertificateRef tp:certId="Esx6_Cert"/>
      <tp:ServerSecurityDetailsRef tp:securityId="Esx6_Security"/>
    </tp:TransportClientSecurity>
  </tp:TransportSender>
  <tp:TransportReceiver>
    <!-- References certificate and URL for the intermediary -->
    <tp:TransportProtocol tp:version="1.1">HTTP</tp:TransportProtocol>
    <tp:Endpoint tp:uri="https://intermediary.example.com:4334/exchange/
        ebxml-intermediary" tp:type="allPurpose"/>
    <tp:TransportServerSecurity>
      <tp:TransportSecurityProtocol tp:version="3.0">SSL</tp:Transport
          SecurityProtocol>
      <tp:ServerCertificateRef tp:certId="ebxml-intermediary_SSL_Server_Cert"/>
```

```
<tp:ClientSecurityDetailsRef tp:securityId="ebxml-intermediary_
SSL_Server_Security"/>
</tp:TransportServerSecurity>
</tp:TransportReceiver>
</tp:Transport>
```

## Extracting 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
```

## Managing CPAs

The **Manage CPAs** link at the bottom of the community summary page opens a page (Figure 168) for viewing details of CPAs that have been imported by a community. You also can use the page for importing CPAs and CPA templates.

### Manage CPAs

Community: *Coronation*

CPA ID	Partner name	Start date	End date	View	
CPA	Gnaraloo	01/01/2006 00:00:00	12/31/2110 00:00:00	<a href="#">HTML</a> <a href="#">XML</a>	<a href="#">Export</a> <a href="#">Delete</a>

Or pick a task

- [Import a CPA](#)
- [Manage CPA templates](#)
- [Clone CPAs](#)

**Figure 168. Manage CPAs page**

On the Manage CPAs page you can:

- View details of an imported CPA by clicking the CPA name (Figure 169). At the bottom of the details page are links for viewing the CPA as HTML and XML and for exporting and deleting. (Clone options are available only for users with peer network licensing.)

### View CPA details

Community: *Coronation*

CPA ID:	CPA
Partner name:	Gnaraloo
Version:	2.0b
Status:	agreed
Start date:	01/01/2006 00:00:00
End date:	12/31/2110 00:00:00
Invocation limit:	0
Concurrent conversations:	0

Allow this CPA to be auto-cloned to peers

**Save changes**

Or pick a task

- [View this CPA as XML](#)
- [View this CPA as HTML](#)
- [Export this CPA](#)
- [Delete this CPA](#)
- [Clone this CPA](#)
- [Manage CPAs](#)

**Figure 169. View CPA details page**

- View a CPA as a HTML document by clicking **HTML**. The **full view** (Figure 170) transforms the CPA XML document into an easy-to-read HTML format. The full view provides details about all possible actions in the CPA. The **MMD view** (Figure 171) displays the meta-data elements required for a back-end system to build all possible MMDs for the CPA.

### Viewing options

Choose how you would like to view the CPA

- Full view  
 MMD view
- 

### General Information

CPA Id: Coronation::Gnaraloo::UBL 2 Small Business Subset Order Notification::id47853  
Status: agreed  
Start date: 2006-01-01T07:00:00Z  
End date: 2110-12-31T07:00:00Z

### Party Information

Party 1 Id: coronation  
Party 1 Id type: string  
Party 2 Id: gnaraloo  
Party 2 Id type: string

### Collaboration Roles: 3

#### Collaboration Role 1

Process specification name: UBL 2 Small Business Subset Order Notification

**Figure 170. CPA full view (only the top of the view is shown due to length)**

**Viewing options**

Choose how you would like to view the CPA

- Full view  
 MMD view
- 

**MMD metadata required to trade using this CPA****Send Order ReceiptAcknowledgement**

From: gnaraloo  
From type: string  
From role: OrderReceiver  
To: coronation  
To type: string  
To role: OrderSender  
Service: Forward Order  
Service type: <http://docs.oasis-open.org/ebxmlbp/ebbp-2.0>  
Action: Send Order ReceiptAcknowledgement

**Send Order ReceiptAcknowledgementException**

From: gnaraloo  
From type: string  
From role: OrderReceiver  
To: coronation  
To type: string  
To role: OrderSender  
Service: Forward Order

**Figure 171. CPA MMD view (only the top of the view is shown due to length)**

- ◆ View a CPA as an XML document by clicking **XML**.
- ◆ Export a CPA to a file or registry by clicking **Export**.
- ◆ Delete a CPA by clicking **Delete**.

The following describes the links at the bottom of the Manage CPAs page:

- ◆ Click **Import a CPA** to import a fully configured CPA that contains the trading details for your community and a partner. CPAs also can be imported automatically by copying the CPA files to [install directory]\[build number]\profiles\autoimport.
- ◆ Click **Manage CPA templates** to import templates that partners can use with the partner registration wizard to build complete CPAs. See [Self-registration of ebXML partners](#) on page 577 for more information.
- ◆ Click **Clone CPAs** to clone CPAs to peer partners. This feature is available only if your user license supports the peer network.

## ***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. The name of the community importing the CPA must be the same in the community profile and the CPA. The community importing the CPA must have a certificate matching the one in the CPA associated with the community profile. 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.

# Tools for CPAs

Command-line tools are available to help manage CPAs. All are found in [install directory]\[build number]\tools and must be run from the tools directory. The following topics describe each tool.

When one of these tools call for entering a file name and path, the path must be in the form of a URL. For example, if the target file is at:

c:\data\cpa.xml

Type the path as:

file:///c:/data/cpa.xml

## ***ebxmlCpaSchematronValidator***

The ebxmlCpaSchematronValidator tool performs tests on the content of a CPA. The tool makes sure matching elements in each PartyInfo element of the CPA are consistent.

Schematron is an XML schema language that can be used to validate XML. For information about Schematron, go to <http://xml.com/> and search for **Schematron**.

Run the tool in the following format:

**ebxmlCpaSchematronValidator [parameter] [path to CPA file]**

The following describes the parameters.

Parameter	Description
-c	URL referencing the CPA to be validated. For example: file:///C:/ebXML/cpa.xml
-f	Force the compilation of the default Schematron rules file.
-h	Prints tool help.
-s	URL for the custom Schematron rules file to compare against the CPA.
-v	Verbose output of the validation processing.

## **ebxmlCpaSecurityGuard**

The ebxmlCpaSecurityGuard tool is used for digitally signing CPAs. Its various functions all relate to signing and verifying digital signatures of a CPA.

The certificates you and your partner use to sign the CPA must be trusted by your community in the trading engine.

The following are example formats for running the script to achieve different results:

### **Sign**

```
ebxmlCpaSecurityGuard -s -n SigningPartyName -x file:///c:/cpatest/certNoPassWord.p12 -c file:///C:/cpaTest/cpaToBeSigned.xml -d C:/cpaTest/output -f oneSigCpa.xml
```

### **Verify**

```
ebxmlCpaSecurityGuard -v -c file:///C:/cpatest/SignedCpa.xml
```

### **Remove last signature**

```
ebxmlCpaSecurityGuard -z -c file:///C:/cpaTest/tempCpaSigned.xml -d C:/cpaTest/output -f oneSigCpa.xml
```

The following describes the parameters.

Parameter (short)	Parameter (long)	Description
-c	--cpa	URL referencing the CPA. For example: file:///C:/ebxml/cpa.xml.
-d	--directory	Location of the output directory.
-f	--file-name	Name of the CPA to be written to the output directory.
-h	--help	Prints tool help.
-i	--info	Prints security information for the CPA.
-n	--party-name	Name of the party signing the CPA. Must match the partyName attribute of the PartyInfo element of the CPA.

Parameter (short)	Parameter (long)	Description
-o	--force	Forces signing or validating the CPA without validating the ds:Reference element in each ProcessSpecification element.
-p	--password	The certificate password. If the certificate does not have a password or has an empty password, omit this option.
-r	--remove-all-signatures	Removes all signatures from the CPA.
-s	--sign	Signs the CPA.
-u	--clean	The same as calling r(remove-all-signatures) and y(remove-all-ds:References) together.
-v	--verify	Verifies the CPA.
-w	--add-all-ds:References	Removes pre-existing ds:Reference and ds:Signature elements and then adds ds:Reference elements to each ProcessSpecification.
-x	--certificate	URL of the certificate that will sign the CPA. For example: file:///C:/ebxml/certs/signingCert.p12.
-y	--remove-all-ds:References	Removes all ds:References elements from each ProcessSpecification in the CPA.
-z	--remove-last-signature	Removes the last signature from the CPA.

## ***ebxmlCpaValidator***

The ebxmlCpaValidator tool performs a schema validation on a CPA.

Run the tool in the following format:

**ebxmlCpaValidator [-offline -online] -cpa [cpa1 file] [cpa2 file]**  
...

The following describes the parameters.

Parameter	Description
-help	Prints tool help.
-offline or -online	Offline: Do not access the Internet to retrieve XML schemas; all required schemas are available locally. This is the default.  Online: Access the Internet to retrieve XML schemas.
-cpa	Paths for one or more CPAs referenced as URLs. For example: file:///tmp/cpa.xml or http://www.server.com/sampleCPA.xml).

## mmdGenerator

The mmdGenerator is for generating all possible MMDs or a specific MMD for a CPA. It can be run from a command line, but also has a user interface option.

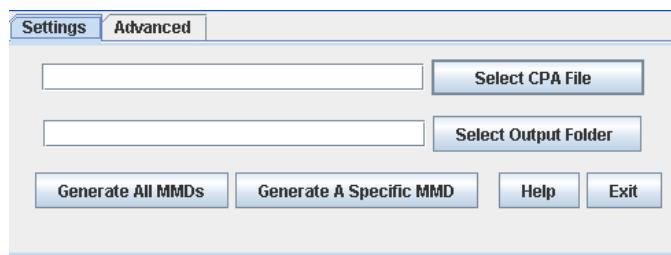
This tool primarily is for testing whether MMDs can be generated correctly from CPAs. But if you use the tool to generate a specific MMD, it could be submitted through integration as a production MMD to the trading engine. If you use the tool to generate all MMDs from a CPA, the generated MMDs could not be used in production unless you manually edit the payload location file path in each MMD.

To see the command-line usage for the tool, use the -h parameter.

Use the following procedure to generate MMDs from the user interface.

### Steps

- 1 To open the mmdGenerator user interface, run the command without parameters or double-click **mmdGenerator** in the tools directory.



**Figure 172. MMD Generator**

- 2** On the Settings tab, click **Select CPA File** to choose a CPA to use to generate MMDs.
- 3** Click **Select Output Folder** to choose the directory to write the generated MMDs.
- 4** On the Advanced tab, you can specify MMD settings for:
  - Synchronous reply
  - Duplicate elimination
  - Signed acknowledgment
  - Acknowledgment requested
- 5** On the Settings tab, select one of the following options for generating MMDs:

#### **Generate All MMDs**

Click to generate all possible MMDs for the CPA. The tool prompts you to select the **from** party. After selecting the party, the tool generates the MMDs and writes the files to the output directory.

#### **Generate A Specific MMD**

Click to generate a single MMD for the CPA. The tool prompts you to type a valid service and action and provide the path to the payload file. Click Generate. The tool then prompts you to select the **from** party. After selecting the party, the tool generates the MMD and writes the file to the output directory.

## **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 for STAR BOD documents earlier than version 3.0. The asterisk indicates the first child element name of the DataArea element.

**DataArea/\***

Action from document location for version 3.0 and later STAR BOD documents. The asterisk indicates the first child element name of the DataArea element.

## 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.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:

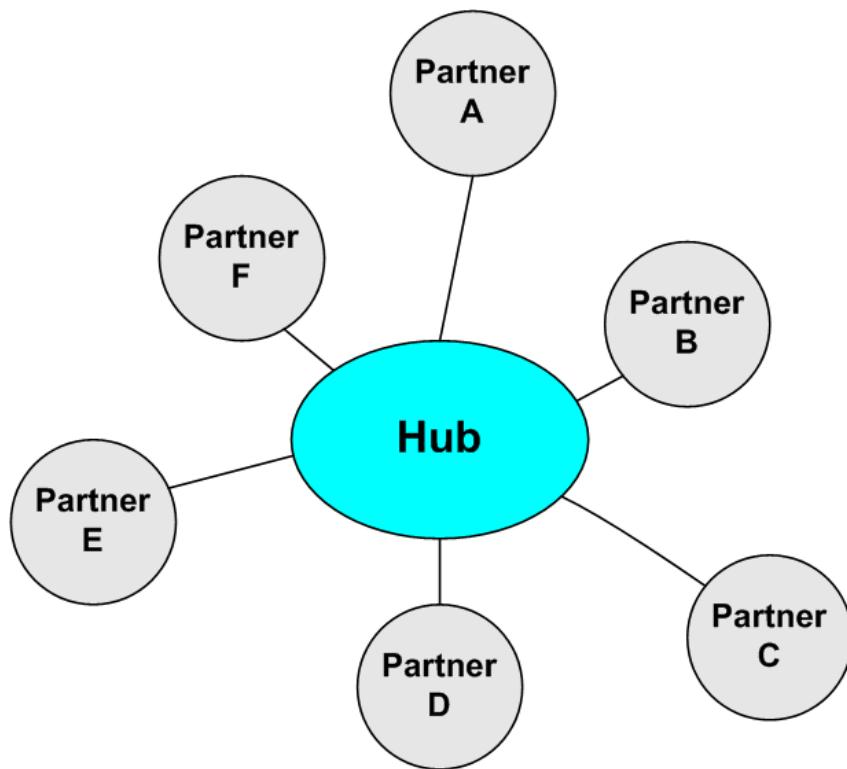
On the maintenance page for the integration pickup exchange, select the Message attributes tab and add an attribute named **CPAId**. Give the attribute a value equal to the CPA ID of the CPA you are using.

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 175.



**Figure 173. 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 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 156.

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 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 Activator 5 or later, install the trading engine (see [Installing and starting the server on page 15](#)) and log on to the user interface (see [Open the user interface on page 22](#)).
- 2 If you use 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 156](#).
- 3 Export your encryption certificate and public key to a file. Include all certificates in the certification path, if possible.

If you use 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 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 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 trade documents successfully with this protocol. For information about RosettaNet see [www.rosettanet.org](http://www.rosettanet.org).

**Note:** RosettaNet requires a special user license to enable the message protocol in the trading engine.

## Concepts

- [RosettaNet overview](#)
- [RosettaNet configuration outline](#) on page 586
- [Add DTD-based PIP](#) on page 588
- [Add schema-based PIP](#) on page 588
- [Configuring pipdefinitions.xml file](#) on page 589
- [RNIF meta-data elements](#) on page 596
- [Message meta-data documents](#) on page 601
- [Special handling of meta-data](#) on page 602

## RosettaNet overview

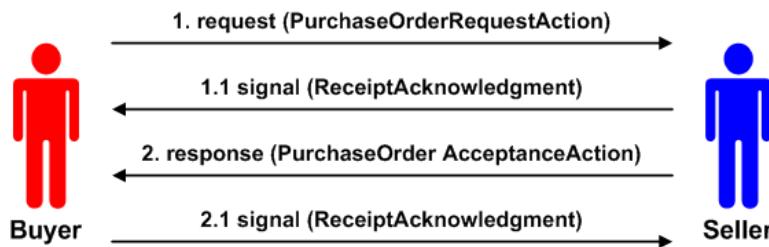
RosettaNet is the name both of a consortium of companies and of a set of processes and standards for conducting electronic business transactions. Activator supports two versions of the RosettaNet Implementation Framework (RNIF). RNIF 1.1 and 2.0 are implementation guidelines for companies that want to create interoperable software that execute Partner Interface Processes (PIPs).

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 Activator. MMDs are interface XML documents that include meta-data in the packaging of RosettaNet headers (preamble, service header, delivery header).

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, schemas 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 174 shows a message flow.



**Figure 174. RosettaNet messages**

Activator does duplicate checking for RosettaNet. The unique identification for a RosettaNet message is the MessageTrackingId. This cannot be disabled in Activator. RosettaNet requires that the PIP instance ID be unique across all PIP instances and that the MessageTrackingId 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 [Installing and starting the server](#) on page 15.
- 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 156.

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 203.

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 206.

If a partner uses Interchange or Activator 4.x, be aware of the following. If the community profile you export has delivery exchanges for both RNIF 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 164.

Make sure the message protocol for sending messages to partners is RosettaNet 1.1 or RosettaNet 2.0.

If a partner uses Interchange or Activator 4.x, the partner profile you import will have delivery exchanges for sending messages for both RNIF 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 PIPs.

For DTD-based PIPs, Add PIP DTDs to the dtds directory. See [Add DTD-based PIP](#) on page 588.

For schema-based PIPs, add PIP schemas to the schemas directory. See [Add schema-based PIP](#) on page 588

- 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 467 or [RosettaNet 2.0 default settings](#) on page 468.
- 8 Define the PIPs in the pipdefinitions.xml file. See [Configuring pipdefinitions.xml file](#) on page 589.

The community now is ready to begin trading messages.

## Add DTD-based PIP

Use this procedure to use a DTD-based PIP not already supported by Activator.

The trading engine supports a number of DTD-based PIPs. The PIP document type definitions (DTDs) are in [install directory]\[build number]\conf\rosettanet\dtds.

### Steps

- 1** Obtain the correct version of the PIP package from RosettaNet at [www.rosettanet.org](http://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]\[build number]\conf\rosettanet\dtds.
- 3** Define the PIP in the pipdefinitions.xml file. See [Configuring pipdefinitions.xml file](#) on page 589.
- 4** When you upgrade later to a newer version of the trading engine, you will have to copy the DTDs to the dtlds directory of the new application. You also will have to change the pipdefinitions.xml file for the new application. See [Planning for upgrades and disaster recovery](#) on page 56.

## Add schema-based PIP

Use this procedure to use a schema-based PIP not already supported by Activator.

The trading engine supports a number of schema-based PIPs. The PIP schemas are in [install directory]\[build number]\conf\rosettanet\schemas.

### Steps

- 1** Obtain the correct version of the PIP package from RosettaNet at [www.rosettanet.org](http://www.rosettanet.org).

A PIP package is a zip file containing a readme file, an XML folder with the machine-readable PIP standard and a descriptive folder with the human-readable form of the PIP standard. The XML folder has schemas defining the structure and allowable content of the messages involved in the PIP.

- 2** Unzip the PIP zip file. Copy the resulting directory to [install directory]\[build number]\conf\rosettanet\schemas.
- 3** Define the PIP in the pipdefinitions.xml file. See [Configuring pipdefinitions.xml file](#) on page 589.
- 4** When you upgrade later to a newer version of the trading engine, you will have to copy the PIP directory to the schemas directory of the new application. You also will have to change the pipdefinitions.xml file for the new application. See [Planning for upgrades and disaster recovery](#) on page 56.

## Configuring 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.

The file is at [install directory]\[build number]\conf\rosettanet.

Figure 175 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.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <PipDefinitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="pipdefinition.xsd" dtdValidation="off" guidelineValidation="off">
- <Pip code="OC2" name="Asynchronous Request-Confirm" version="R01.02">
- <Activity name="Asynchronous Test Request" retryAttempts="3">
  - <Action name="Asynchronous Test Request Action"
    dtd="OC2_MS_R01_02_AsynchronousTestRequest.dtd"
    rootElement="PipOC2AsynchronousTestRequest" fromService="Initiator Service"
    toService="Responder Service">
    <Signal timeToAcknowledge="5" />
  </Action>
</Activity>
- <Activity name="Asynchronous Test Confirmation" retryAttempts="3">
  - <Action name="Asynchronous Test Confirmation Action"
    dtd="OC2_MS_R01_02_AsynchronousTestConfirmation.dtd"
    rootElement="PipOC2AsynchronousTestConfirmation" fromService="Responder Service"
    toService="Initiator Service">
    <Signal timeToAcknowledge="5" />
  </Action>
</Activity>
</Pip>
- <Pip code="3B13" name="Shipping Order Confirmation Notification" version="V01.01">
- <Activity name="Notify of Shipment Confirmation" retryAttempts="3">
  - <Action name="Shipping Order Confirmation Notification Action"
    dtd="3B13_MS_V01_01_ShippingOrderConfirmationNotification.dtd"
    rootElement="Pip3B13ShippingOrderConfirmationNotification" fromService="Initiator Service"
    toService="Responder Service">
    <Signal timeToAcknowledge="15" />
  </Action>
</Activity>
</Pip>
- <Pip code="3B3" name="Distribute Shipment Status" version="V11.00">
- <Activity name="Shipment Status Distribution" retryAttempts="3">
  - <Action name="Shipment Status Distribution Action" dtd="ShipmentStatusDistribution_01_02.xsd"
    rootElement="ShipmentStatusDistribution" fromService="Transport Service Provider Service"
    toService="In-transit Information User Service" fromRole="Transport Service Provider" toRole="In-transit Information User" globalDocumentFunctionCode="Request"
    fromGlobalPartnerClassificationCode="Carrier" toGlobalPartnerClassificationCode="Distributor">
    <Signal timeToAcknowledge="120" />
  </Action>
</Activity>
</Pip>
</PipDefinitions>
```

**Figure 175. Example of pipdefinitions.xml file**

**Note:** The trading engine supports asynchronous responses, but not synchronous responses for RNIF.

The following describes the elements in the pipdefinitions.xml file. See Figure 175 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]\[build number]\conf\rosettanet\dtds or the schema to [install directory]\[build number]\conf\rosettanet\schemas.

**guidelineValidation**

For DTD-based PIPs only, indicates (**on** or **off**) whether the trading engine validates the document against a guidelines XML file. If **on**, copy the file to [install directory]\[build number]\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 **oC2**.

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.

For DTD-based PIPs, 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.

For schema-based PIPs, if dtdValidation="on", the value is the name of the schema to validate against. The name is available in [install directory]\[build number]\conf\rosettanet\schemas. Look in the XML\Interchange directory for the desired PIP.

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.

**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.

**fromRole**

An attribute of the Action element, this is the role the trading partner who sends the message plays in the PIP. This value is specified in the RosettaNet PIP specification document. This is required to build the RosettaNet headers.

The value is parsed from the service content for DTD-based PIPs. However, the value in pipdefinitions.xml is used only when the value is not found in the service content.

The value is required for schema-based PIPs. It is not specified in the service content of schema-based PIPs.

**toRole**

An attribute of the Action element, this is the role the trading partner who receives the message plays in the PIP. This value is specified in the RosettaNet PIP specification document. This is required to build the RosettaNet headers.

The value is parsed from the service content for DTD-based PIPs. However, the value in pipdefinitions.xml is used only when the value is not found in the service content.

The value is required for schema-based PIPS. It is not specified in the service content of schema-based PIPs.

**globalDocumentFunctionCode**

This is an attribute of the Action element. The possible values are:

**Request.** The business document is a request for a business action to be performed by a partner.

**Response.** The business document is a response to a requesting partner.

This is required to build RNIF 1.1 headers. The value is ignored when packaging RNIF 2.0 messages. This attribute is ignored for DTD-based PIPs. The value is required for schema-based PIPS when using RNIF 1.1 packaging.

**fromGlobalPartnerClassificationCode**

An attribute of the Action element, this is a classification of the role the trading partner who sends the message plays in this PIP. The valid values are specified in the RosettaNet Service Header Message Guidelines for RNIF 1.1 and RNIF 2.0. This is required to build the RosettaNet headers.

The value is parsed from the service content for DTD-based PIPs. However, the value in pipdefinitions.xml is used only when the value is not found in the service content.

The value is required for schema-based PIPS. It is not specified in the service content of schema-based PIPs.

**toGlobalPartnerClassificationCode**

An attribute of the Action element, this is a classification of the role that the trading partner receiving the message plays in this PIP. The valid values are specified in the RosettaNet Service Header Message Guidelines for RNIF 1.1 and RNIF 2.0. This is required to build the RosettaNet headers.

The value is parsed from the service content for DTD-based PIPS. However, the value in pipdefinitions.xml is used only when the value is not found in the service content.

The value is required for schema-based PIPS. It is not specified in the service content of schema-based PIPs.

**PipInstanceId**

Used to specify when special handling of RosettaNet meta-data within the service content is enabled. This functionality aids in the trading of dual-action PIPs when MMDs are not used.

**parseFromServiceContent**

Enables or disables parsing and writing pipInstanceId, ReplyToTrackingId and Global Usage Code from or to service content. Defaults to false.

**separator**

This attribute is used to specify the separator character to delineate pipInstanceId, ReplyToTrackingId and Global Usage Code when reading from or writing to the service content. The default separator character is a colon ( : ).

**xpath**

This attributes specifies the XPath expression to parse or write pipInstanceId, ReplyToTrackingId and Global Usage Code from or to.

For DTD-based service content, defaults to:

```
/*/thisDocumentIdentifier/ProprietaryDocumentIdentifier for  
DTD based service contents.
```

For schema-based service content, defaults to:

```
/*/ssdh:DocumentHeader[1]/ssdh:DocumentInformation[1]/  
ssdh:DocumentIdentification[1]/ssdh:Identifier[1]
```

The XPath must include namespace prefixes for schema-based service contents.

### **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 600.

#### **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 GlobalBusinessIdentifier in the service content in this hierarchy:

#### **DTD**

toRole

PartnerRoleDescription

PartnerDescription

BusinessDescription

GlobalBusinessIdentifier

**Schema**

sshd:DocumentHeader  
sshd:Receiver  
upi:PartnerIdentification  
udt:[\*1]

**SenderRoutingId**

The ID of the sending partner. This must be a DUNS number.  
Optional.

If not specified, the value comes from the GlobalBusinessIdentifier in the service content in this hierarchy:

**DTD**

fromRole  
PartnerRoleDescription  
PartnerDescription  
BusinessDescription  
GlobalBusinessIdentifier

**Schema**

sshd:DocumentHeader  
sshd:Sender  
upi:PartnerIdentification  
udt:[\*1]

**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**

For RNIF 1.1 only, a unique identifier for a specific instance of this transaction within a specific instance of the process. Actions within the same instance of a transaction must have the same value. This value is from the service header.

## MMD elements

The following elements are applicable only to MMDs.

For MMDs, some names of meta-data elements are slightly different than those listed in [Description of elements](#) on page 596, but the usage is the same. The following lists these discrepancies.

### **RemovePayloadAfterProcessing**

Indicates (**true** or **false**) whether the trading engine deletes the payload from the file system after processing the message. (MMDs always are deleted after processing.)

Use of this element is optional. If not used, the payload is not deleted, which is the same behavior as using the element with a value of **false**. If RemovePayloadAfterProcessing is **true**, payloads are deleted after being picked up.

This also works for the resubmit case in which payloads are retrieved from the backup directory.

### **ToRoutingId**

Same as [ReceiverRoutingId](#) on page 598.

### **FromRoutingId**

Same as [SenderRoutingId](#) on page 599.

### **ToClassificationCode**

Same as [ToGlobalPartnerClassificationCode](#) on page 600.

### **FromClassificationCode**

Same as [FromGlobalPartnerClassificationCode](#) on page 596.

### **KnownInitiatingRoutingId**

Same as [InitiatingRoutingIdKnown](#) on page 597.

## **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 176 is an example of an MMD associated with a RosettaNet document. For meta-data descriptions, see [RNIF meta-data elements](#) on page 596.

```
- <MessageMetadataDocument id="metadataID51736371109613218815JAPPLLEGATE-T40"
  protocol="RosettaNet" protocolVersion="2.0">
  - <!--
        <Metadata name="PipInstanceId">12345678901</Metadata>
    -->
  <Metadata name="PipCode">3B13</Metadata>
  <Metadata name="ToService">Provider Service</Metadata>
  <Metadata name="FromService">Initiator Service</Metadata>
  <Metadata name="BusinessActivityIdentifier">Shipping Order Confirmation Notification</Metadata>
  <Metadata name="GlobalBusinessActionCode">Shipping Order Confirmation Notification
Action</Metadata>
  <Metadata name="GlobalUsageCode">Test</Metadata>
  <Metadata name="PipVersion">V01.01</Metadata>
  - <MessagePayloads>
    - <Payload id="service-content">
      <RemovePayloadAfterProcessing>true</RemovePayloadAfterProcessing>
      - <Location type="filePath">
          /Users/mmdIntegration/remerry-mif2esx6-mif-svccontent.xml
        </Location>
      </Payload>
    </MessagePayloads>
  </MessageMetadataDocument>
```

**Figure 176. 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, 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 177). For outbound integration the GlobalUsageCode and PipInstanceId can be included in the MMD. For response messages of dual-action PIPs, the PipInstanceId and InReplyToTrackingId also must be included in the MMD. If included within the MMD, the GlobalUsageCode, PipInstanceId and InReplyToMessageId values from the MMD are used in the packaged RNIF message.

```
- <MessageMetadataDocument id="ID92102651105484506795birddog-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="ID69561991105484506795birddog-mac.local">
    - <Location type="filePath">
      ./data/in/RN_Service_Content_1105484499336_978_BirdDog_7
    </Location>
  </Payload>
</MessagePayloads>
</MessageMetadataDocument>
```

Figure 177. MMD produced for inbound integration

## Without MMDs

If MMDs are not used, the outbound service content can be parsed to determine the PipInstanceId, GlobalUsageCode, InReplyToTrackingId and InitiatingRoutingId to use in the packaged document.

By default this functionality is disabled, but can be enabled in pipdefinitions.xml. See [Configuring pipdefinitions.xml file](#) on page 589.

For response messages of dual-action PIPs the PipInstanceId and InReplyToTrackingId must be included in the service content. If included, the GlobalUsageCode, PipInstanceId, InReplyToTrackingId and InitiatingRoutingId values from the service content are used in the packaged RNIF message.

## DTD-based service contents

When enabled, the following field is parsed:

```
thisDocumentIdentifier/ProprietaryDocumentIdentifier
```

The location to parse can be modified in pipdefinitions.xml (Figure 178).

```
<thisDocumentIdentifier>
  <ProprietaryDocumentIdentifier>
    [proprietary document ID]:[GlobalUsageCode]
    PipInstanceId:[InReplyToTrackingId]:[InitiatingRoutingId]
  </ProprietaryDocumentIdentifier>
</thisDocumentIdentifier>
```

**Figure 178. DTD-based service contents**

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 include:

- GlobalUsageCode second
- PipInstanceId third
- InReplyToTrackingId fourth
- InitiatingRoutingId fifth

Blank fields are acceptable within the string. For example, to set only the proprietary document ID and PipInstanceId the string would look like:

**Myid::12346**

Where **Myid** is the proprietary document ID and **12346** is the PipInstanceId.

The GlobalUsageCode, PipInstanceId, InReplyToTrackingId and InitiatingRoutingId values from the outbound message are used in building the protocol headers in the packaged RNIF message.

Note that the PipInstanceId, GlobalUsageCode, InReplyToTrackingId and InitiatingRoutingId values are stripped from the service content that is packaged and produced to the trading partner. The identifier value is included in the packaged service content.

If enabled, the PipInstanceId, GlobalUsageCode, InReplyToTrackingId and InitiatingRoutingId are written to the received service content produced to integration. This enables back-end systems to know the values necessary for sending response documents for dual-action PIPs.

## Schema-based service contents

If enabled, the following field is parsed:

```
/*/ssdh:DocumentHeader[1]/ssdh:DocumentInformation[1]/
ssdh:DocumentIdentification[1]/ssdh:Identifier[1]
```

The location can be changed in pipdefinitions.xml (Figure 179).

---

```
<ssdh:DocumentIdentification schemaVersion="">
  <ssdh:Identifier>
    [identifier]:[GlobalUsageCode]:[PipInstanceId]:[InReplyToTrackingId]
    :[InitiatingRoutingId]
  </ssdh:Identifier>
  <ssdh:StandardDocumentIdentification schemaVersion="">
    <ssdh:Standard>RosettaNet</ssdh:Standard>
    <ssdh:Version>PIP3B3v11.00</ssdh:Version>
  </ssdh:StandardDocumentIdentification>
</ssdh:DocumentIdentification>
```

---

**Figure 179. Schema-based service contents**

The value of the Identifier element should take the form of a colon-separated list of strings. The first string is the identifier. Optionally include:

- GlobalUsageCode second
- PipInstanceId third
- InReplyToTrackingId fourth
- InitiatingRoutingId fifth

Blank fields are acceptable within the string. For example, to set only the identifier and pipInstanceId the string would look like:

**Myid::12346**

Where **Myid** is the identifier value and **12346** is the PipInstanceId.

The GlobalUsageCode, PipInstanceId, InReplyToTrackingId and InitiatingRoutingId values from the outbound message are used in building the protocol headers in the packaged RNIF message.

Note that the PipInstanceId, GlobalUsageCode, InReplyToTrackingId and InitiatingRoutingId values are stripped from the service content that is packaged and produced to the trading partner. The identifier value are included in the packaged service content.

If enabled, the PipInstanceId, GlobalUsageCode, InReplyToTrackingId and InitiatingRoutingId are written to the received service contents produced to integration. This allows back-end systems to know the values necessary for sending response documents for dual-action PIPs.



# 29 Synchrony CSOS

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

- ◆ [Overview of CSOS functionality](#)
- ◆ [How it works on page 608](#)
- ◆ [CSOS configuration for sending on page 609](#)
- ◆ [CSOS configuration for receiving on page 610](#)

## Procedure

- ◆ [Import CSOS signing certificate on page 611](#)
- ◆ [CSOS certificate revocation lists on page 612](#)
- ◆ [Identify CSOS purchase orders on page 612](#)
- ◆ [Sign pending orders on page 625](#)

## 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.

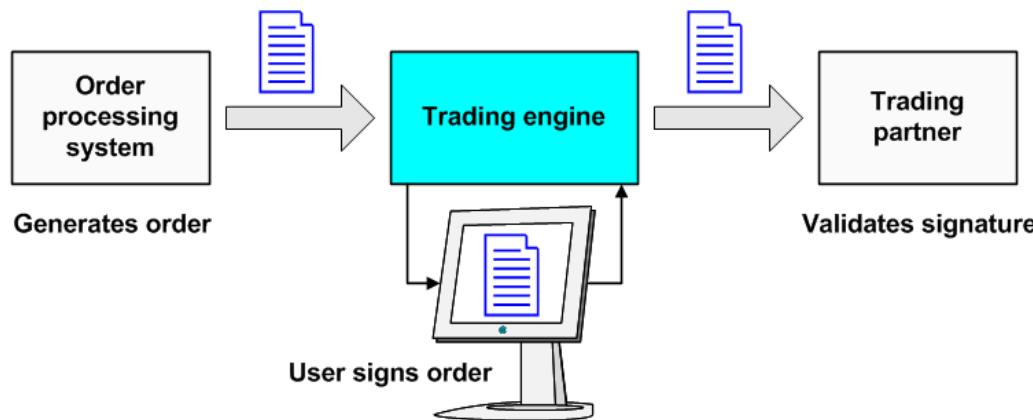
When the trading engine validates the signature of received orders, it checks the signer's certificate against a DEA list to make sure the certificate has not been revoked. This is done using a certificate revocation list (CRL). The CRL is issued and updated by DEA. The trading engine has the ability 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 180 illustrates how CSOS functionality works in the trading engine according to the following scenario.



**Figure 180. 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. A system administrator performs the steps, except as noted when a CSOS signing user should perform a task.

- 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

[Installing and starting the server](#) on page 15

[Getting started with Activator](#) on page 147

- 2 Select **Users and roles > Add a role** in the user interface.

- 3 Type the name for the role. This is the role to assign to the users who log on to view CSOS orders and sign orders using a DEA-issued signing certificate.

At minimum, select the permission **Approve CSOS orders** for the role. You can grant other permissions, depending on how much latitude you want to give to CSOS order signers. You may not want to give the role the same permissions as the administrator role. For more information see [Role permissions](#) on page 80.

Click **Add this role** to create the role.

- 4 Select **Users and roles > Add a user**.

- 5 Add the user account for the person who will log on and sign CSOS purchase orders with a signing certificate. Assign the user to the CSOS role you added in step 3.

Click **Add this user** to create the user.

Repeat this step if you need to add another CSOS signing user.

- 6 Have the CSOS user you added in step 5 log on to the user interface.
- 7 Have the CSOS user select **Change my user account** on the toolbar. Depending on the permissions for the role assigned to the user, this link is on the **User settings** menu or the **Users and roles** menu.
- 8 Have the CSOS user select the personal certificates tab and click **Add a certificate** to open the certificate wizard.

The certificate to add is the user's signing certificate from DEA. A signing certificate is needed to send signed orders to a partner.

Import the signing certificate to the trading engine. See [Import CSOS signing certificate](#) on page 611.

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).

- 9 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 612.
- 10 Display and approve the orders in the user interface. See [Sign pending orders](#) on page 625.

## 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
- [Installing and starting the server](#) on page 15
- [Getting started with Activator](#) on page 147

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 612](#).
- 3** Configure the trading engine to identify CSOS orders received from partners. See [Identify CSOS purchase orders on page 612](#).

## 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 **Change my user account** on the toolbar. Depending on the permissions for the role assigned to the user, this link is on the **User settings** menu or the **Users and roles** menu.
- 2** Select the personal certificates tab and click **Add a certificate** to open the certificate wizard.
- 3** Type the path to the certificate file or click **Browse** to locate the file.
- 4** In the Password field under the Certificate file field, type the password the DEA provided. This is the password that protects the certificate file.
- 5** In the Certificate password field, type the password you want to use. This password supersedes the password provided by DEA. This is the password you will use when you sign a CSOS order. Also type the new password in the Confirm password field.
- 6** Click **Next**.

A name for the certificate is displayed. You can use this name or type a different name.

- 7 Click **Finish** to complete importing 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 437 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 these tabs:

- Order identification tab

- ◆ Order sources tab
- ◆ Related documents tab
- ◆ CSOS duplicate orders tab

If you send or receive signed orders, you must use the order identification 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 order identification 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.

## ***Order identification tab***

The order identification tab of the identify CSOS purchase orders page (Figure 181) 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

<a href="#">Order identification</a>	<a href="#">Order sources</a>	<a href="#">Related documents</a>	<a href="#">CSOS duplicate orders</a>																		
<p>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.</p> <p><b>EDI documents</b></p> <p>Identify EDI 850s as CSOS purchase orders by entering an XPath that locates a segment or element value. EDI documents will temporarily be converted to XML as part of the identification process.</p> <table border="1"> <thead> <tr> <th>Identity XPath</th> <th>Attribute XPaths</th> <th>Document type</th> </tr> </thead> <tbody> <tr> <td>No EDI document types are defined. <a href="#">Add an EDI document identifier</a> or <a href="#">Accept all EDI 850s as CSOS purchase orders</a></td> <td></td> <td></td> </tr> <tr> <td colspan="3"><input checked="" type="checkbox"/> <a href="#">Add an EDI document identifier</a></td> </tr> </tbody> </table> <p><b>XML documents</b></p> <p>Identify XML documents as CSOS purchase orders by entering an XPath that locates an element, attribute or text string.</p> <table border="1"> <thead> <tr> <th>Identity XPath</th> <th>Attribute XPaths</th> <th>Document type</th> </tr> </thead> <tbody> <tr> <td>No XML document types are defined. <a href="#">Add an XML document identifier</a></td> <td></td> <td></td> </tr> <tr> <td colspan="3"><input checked="" type="checkbox"/> <a href="#">Add an XML document identifier</a></td> </tr> </tbody> </table>				Identity XPath	Attribute XPaths	Document type	No EDI document types are defined. <a href="#">Add an EDI document identifier</a> or <a href="#">Accept all EDI 850s as CSOS purchase orders</a>			<input checked="" type="checkbox"/> <a href="#">Add an EDI document identifier</a>			Identity XPath	Attribute XPaths	Document type	No XML document types are defined. <a href="#">Add an XML document identifier</a>			<input checked="" type="checkbox"/> <a href="#">Add an XML document identifier</a>		
Identity XPath	Attribute XPaths	Document type																			
No EDI document types are defined. <a href="#">Add an EDI document identifier</a> or <a href="#">Accept all EDI 850s as CSOS purchase orders</a>																					
<input checked="" type="checkbox"/> <a href="#">Add an EDI document identifier</a>																					
Identity XPath	Attribute XPaths	Document type																			
No XML document types are defined. <a href="#">Add an XML document identifier</a>																					
<input checked="" type="checkbox"/> <a href="#">Add an XML document identifier</a>																					

**Figure 181. Order identification tab of Identify CSOS purchase orders page**

Using the order identification 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 source 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. Source XML documents 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 identifier**. This displays a section (Figure 182) 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.

Identity XPath	Attribute XPaths	Document type
	DEA Number: <input type="text"/> ...	Order Number: <input type="text"/> ...
		<input type="button" value="Save"/> <input type="button" value="Cancel"/>

**Figure 182. 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 183) 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.

Identity XPath	Attribute XPaths	Document type
Accepting all EDI 850s	DEA Number: <input type="text"/> ...	<input type="button" value="Save"/> <input type="button" value="Delete"/> <input type="button" value="Set identity XPath"/> <input type="button" value="Test"/>
	Order Number: <input type="text"/> ...	

**Figure 183. 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 184, 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 o3X123456 in the second position of the REF segment.

---

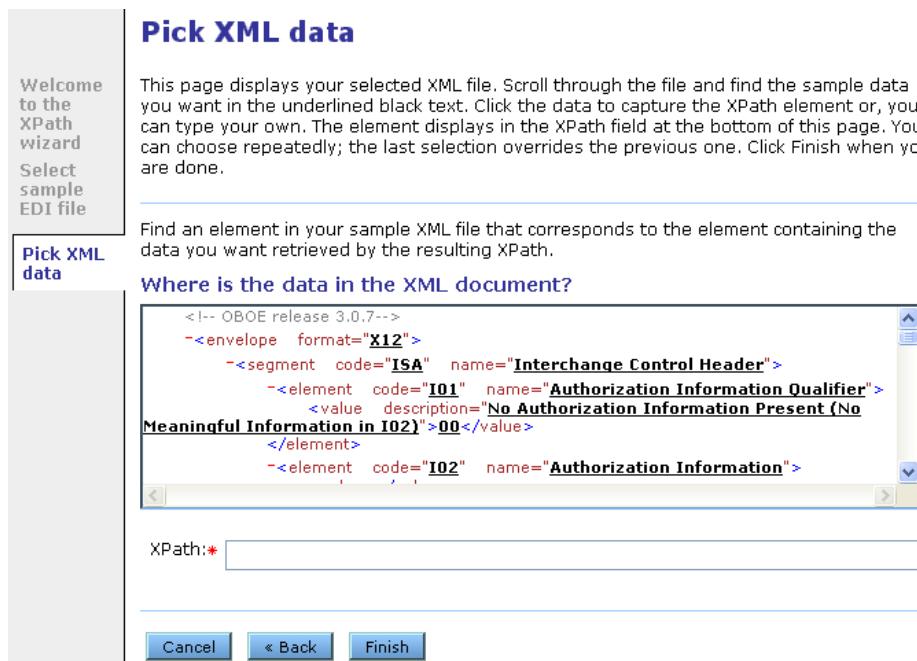
```
ISA*00*          *00*          *ZZ*PURCHASER      *ZZ*SUPPLIER
*030128*0647*U*00400*000002122*0*P*|~GS*PO*177667227*8006670959*20030128
*06474500*2121*X*004010~ST*850*0001~BEG*00*SA*581020016**20030128~REF*D1
*03X123456~FOB*PP~CSH*N~DTM*002*20030205~N1*SU*BAXTER
ACC*11*CD1234567~N1*ST*ANY DISTRIBUTOR*11*BA2893611~N3*1234 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 TUBE
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 184. Sample 850 document**

The XPath wizard converts the EDI into a much longer XML document. See [EDI to XML conversion example](#) on page 628 to view the converted document.

The wizard displays the converted document on the pick XML data page (Figure 185). Using your browser's search feature, search for a value (for example, the order number o3X123456). When the value is found, click it to generate the XPath in the XPath field.

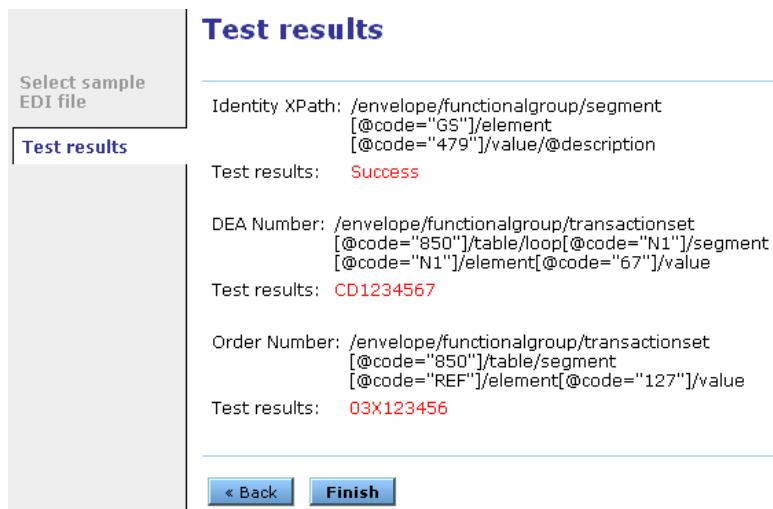


**Figure 185. Pick XML data page in XPath wizard**

Using the values desired for the sample 850 in Figure 184, 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 order identification tab to check whether the XPaths are correct and yield the desired data. Figure 186 shows the test results for the preceding XPaths.

**Figure 186. XPath test results**

Recall that the DEA registration number desired from the sample 850 in Figure 184 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 187 shows the test results again after changing the DEA number XPath. This time the desired DEA number is produced.

The screenshot shows a user interface for testing EDI documents. On the left, a sidebar has a 'Select sample EDI file' button and a 'Test results' button (which is highlighted). The main area is titled 'Test results' and contains three entries:

- 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

At the bottom are 'Back' and 'Finish' buttons.

**Figure 187. 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 511.

## Identifying XML documents

To identify XML documents for CSOS handling, click **Add an XML document identifier**. This displays a section (Figure 188) 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.

Identity XPath	Attribute XPaths	Document type
<input type="text"/> <input type="button" value="..."/>	DEA Number: <input type="text"/> <input type="button" value="..."/> Order Number: <input type="text"/> <input type="button" value="..."/>	<input type="button" value="Save"/> <input type="button" value="Cancel"/>

**Figure 188. 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 511.

## **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 order identification 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 order identification 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

[Order identification](#) [Order sources](#) [Related documents](#) [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. <a href="#">Add an order source</a>		

**Figure 189. 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

[Order identification](#) [Order sources](#) [Related documents](#) [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	Save	Cancel
from <input type="button" value="any exchange"/>	<input type="button" value="any party"/>	<input type="button" value="any party"/>	<input type="button" value="Save"/>	<input type="button" value="Cancel"/>

**Figure 190. Default order sources settings**

This default choice means:

### If sending orders

All purchase order documents defined on the order identification tab are picked up from any integration exchange and queued for signing.

### If receiving orders

All purchase orders defined on the order identification 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 tab. 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 order identification 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.

## Related documents tab

The related documents tab of the identify CSOS purchase orders page lets you identify documents in the backup directory and database records that need to be retained for as long as CSOS orders. The DEA requires a minimum retention of two years for CSOS orders and related documents. By default, the trading engine is configured to retain documents identified

as CSOS orders for two years. This tab lets you identify documents such as invoices and ship notices that must be retained along with the related CSOS orders.

## Identify CSOS purchase orders

Identity XPath/Document number	Attribute XPaths	Document type
No EDI document types are defined. <a href="#">Add an EDI document identifier</a>		
<input checked="" type="checkbox"/> <a href="#">Add an EDI document identifier</a>		

Identity XPath	Attribute XPaths	Document type
No XML document types are defined. <a href="#">Add an XML document identifier</a>		
<input checked="" type="checkbox"/> <a href="#">Add an XML document identifier</a>		

**Figure 191. Related documents tab of identify CSOS purchase orders page**

The related documents tab is similar to the order identification tab. An important difference is: When you click **Add an EDI document identifier**, you can type an EDI document type number in the Identity XPath/Document number field rather than specify an XPath. This tells the trading engine that all EDI documents of that type are CSOS related.

When identifying related documents, specifying XPaths for a DEA number and order number are optional. But doing so helps to link CSOS orders and related documents in Message Tracker.

See [Order identification tab on page 613](#) for guidance for using the related documents tab.

The age of messages and database records to purge normally is set on a global purge configuration page in the user interface. CSOS orders and related documents have a minimum expiration of two years, regardless whether the global purge age is less than two years. If the global purge age is set to more than two years, CSOS orders and related documents are retained for the longer period.

For information about global purge configuration, see [Delete unwanted files and records on page 518](#).

## CSOS duplicate orders tab

The CSOS duplicate orders tab of the identify CSOS purchase orders page (Figure 192) lets you specify whether to accept or reject duplicate CSOS orders.

### Identify CSOS purchase orders

The screenshot shows a web-based configuration interface for identifying CSOS purchase orders. At the top, there are four tabs: 'Order identification', 'Order sources', 'Related documents', and 'CSOS duplicate orders'. The 'CSOS duplicate orders' tab is currently selected, indicated by a blue border. Below the tabs, the text 'Community: Acme Industries' is displayed. Under this, there are two radio buttons: one selected ('Reject duplicate CSOS purchase orders') and one unselected ('Allow duplicate CSOS purchase orders'). At the bottom of the panel, there is a link labeled 'Add an exception for a partner'.

**Figure 192. 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 193).

### 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

[View past orders](#)  
 [Configure order sources](#)

**Figure 193. CSOS orders to approve page**

- 2** Click the date of a pending order to display it on the approve CSOS order page (Figure 194).

### 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.	Name and Address of Registrant			
BA2893611	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.OFFICIAL ORDER FORMS -SCHEDULES I & II DRUG ENFORCEMENT ADMINISTRATION SUPPLIER'S COPY 1			39558785

Signing certificate: [cert 1 Department of Justice](#)

Password:

**Figure 194. 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 195). 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.

[View pending orders](#)  
 [View past orders](#)  
 [Configure order sources](#)

**Figure 195. 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 196). 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:  
\_\_\_\_\_

[View pending orders](#)  
 [View past orders](#)  
 [Configure order sources](#)

**Figure 196. 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.

# 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 614.

Figure 197 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 197. 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>
</segment>
-<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>
</segment>
-<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>
</segment>
</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>
</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 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>
</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 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>
</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 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>
</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>
-<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>
```



# 30 Transport guidelines

The following topics provide information about exchanging messages with partners via some popular transport methods.

## Concepts

- [Secure HTTP solutions](#)
- [Troubleshooting HTTP connections](#) on page 641
- [Trading large messages](#) on page 643

## Secure HTTP solutions

When using HTTP-based message protocols — AS2, secure file, ebXML, web services, Rosettanet — security considerations are similar to hosting a web server. There are industry-standard solutions to make this type of communication secure. Solutions recommended for Activator are described in the following topics.

### *Embedded HTTP server*

You can configure Activator to accept incoming HTTP connections directly from trading partners by adding an inbound delivery exchange that uses the embedded HTTP server. The following is a list of safety options to use in conjunction with the embedded server. The options are numbered from lowest to highest security.

#### Option 1. Message-level security

It is safe to allow inbound connections directly to the embedded HTTP server if your message validation rules require messages (payloads) to be signed and encrypted (the default). This is true even if the trading engine is running on a machine within your protected network, because the embedded server will reject any message that was not signed with the partner's private key.

This level of security is sufficient for most applications.

**Advantage:** High level of security with minimal per-partner maintenance overhead.

**Disadvantage:** Does not protect against denial-of-service attacks.

## Option 2. Firewall filtering

An additional level of security can be achieved by configuring rules on your firewall to allow only inbound connections from known-partner IP addresses. You also can allow connections only to designated ports (for example, 4080 and 1443).

This level of security, in combination with requiring signed and encrypted payloads, is very high and should be sufficient for virtually all applications.

**Advantage:** Protects against denial-of-service attacks.

**Disadvantage:** IP address filtering requires updating firewall rules when a partner is added or removed.

## Option 3. SSL

If you will be receiving any unencrypted messages, you should require your partners to connect using HTTPS (SSL) so no one on the Internet can eavesdrop on the contents.

**Advantage:** Eavesdroppers will be thwarted even if messages are not encrypted.

**Disadvantage:** Extra CPU load introduced by SSL (but see option 5). The user cannot configure the encryption algorithm for HTTPS. It is better to require message-level security because the encryption algorithms are stronger.

## Option 4. Client-authenticated SSL

With SSL, an additional level of security called client authentication can be configured. This option is selected by default when you define an SSL embedded server in the trading engine. Client authentication requires your partner to present a certificate before the inbound connection can be accepted. This is similar to the concept of requiring payloads to be signed.

If you choose to allow unsigned payloads, it is highly recommended that you require client-authenticated SSL to prevent connections from being accepted from unknown parties.

**Advantage:** Unwanted connections are refused as early as possible, before any part of the payload is received.

**Disadvantage:** Within the trading engine you must maintain a list of trusted SSL client-authentication certificates for each partner. This is in addition to the list of partner certificates you must maintain for encrypting messages and verifying signatures.

## Option 5. Hardware SSL

Options 3 and 4 describe SSL and client-authenticated SSL in terms of the embedded HTTPS server. But the best performance and highest security can be achieved with a hardware device in the DMZ to terminate SSL connections.

Your existing load balancer may be able to do this or you may be able to purchase an add-on SSL module. Alternatively, some vendors offer SSL accelerator cards that can be installed in a server to terminate SSL connections.

**Advantage:** Off-loads CPU-intensive SSL processing from the trading engine to the hardware device. Connections are refused in the DMZ before they reach the trading engine. Very high resistance to denial-of-service attacks.

**Disadvantage:** Requires external hardware device or card. If client-authenticated SSL is used, requires a list of trusted SSL client-authentication certificates to be maintained outside of the trading engine using software included with your hardware device.

## Summary of options

In summary, most users will chose one of the following levels of security when employing the embedded HTTP server.

### High security

The default behavior of Activator in requiring payloads to be signed and encrypted affords a very high level of security right out of the box, even if connections are terminated within your private network by the embedded HTTP server.

This level employs the security described in option 1.

### High security and denial-of-service resistance

Adding IP and port filtering rules to your firewall will prevent connections from being accepted even before the payload has been examined.

This level combines the security described in options 1 and 2.

### Maximum security

The highest possible interoperable level of security can be achieved by using firewall rules in combination with client-authenticated SSL, terminated by a hardware device in your DMZ. In this case, payloads do not need to be signed or encrypted, but can be if an additional level of security is desired.

This level combines the security described in options 1 and 2 in addition to either option 4 or option 5.

## ***External staged HTTP server***

A staged HTTP server usually resides in your DMZ, accepts incoming connections and stages incoming messages to disk. A servlet installed on the external HTTP server allows the trading engine to poll for inbound files that have been dropped off by partners. Functionally, the interaction of Activator with a staged HTTP server is similar to the way it interacts with an FTP server. Commonly used HTTP servers include Tomcat, WebLogic and WebSphere.

The following topics describe advantages and disadvantages to this transport.

### **Advantages**

- 1** Some argue that an external HTTP server provides better security than the embedded server. This is because the external server terminates inbound connections from the Internet in the DMZ. It also does not allow any inbound connections all the way through to the protected network, since Activator polls the embedded server for new files.

However, the better security argument is more a matter of perception than fact, as state-of-the-art levels of security can be achieved with the embedded HTTP server, as described in [Embedded HTTP server](#) on page 637.

- 2** If the trading engine is shut down, inbound messages will queue on the external server, allowing maintenance to be performed without causing partners to experience connection errors. However, it should be noted that partners generally would employ retries to deal with connection errors. Also, using a Activator cluster will achieve a similar result since one node can be serviced while another continues to do work.

- 3 Under peak load conditions messages will queue on the external server if the trading engine is too busy to process them, reducing the chances of partners receiving 503 (busy) errors.

## Disadvantages

- 1 Some argue that staging the files to a file system that is accessible from the DMZ introduces security risks. However, this argument is questionable if the payloads are encrypted. After all, the encryption was good enough when the files were passing through the Internet.
- 2 Having to poll the external server introduces latency that is not present with the embedded server.
- 3 Adding an external server increases the total complexity of the system, increasing the number of things that could break or introducing security holes.

# Troubleshooting HTTP connections

The following flow charts illustrate an outbound HTTP connection, what can go wrong, and how you can resolve connection issues. In addition, if your community is on the receiving end of a partner's outbound connection, the charts can help you resolve issues on your side of the connection.

These charts also apply to HTTPS (HTTP over Secure Sockets Layer). HTTPS uses an SSL certificate to encrypt the connection. Message traffic passing back and forth through the encrypted SSL tunnel is pure HTTP.

HTTP is a popular transport available under many message protocols, including AS2, secure file, ebXML, web services, RosettaNet, and the no packaging protocol. For more information see [Trading delivery exchanges](#) on page 203.

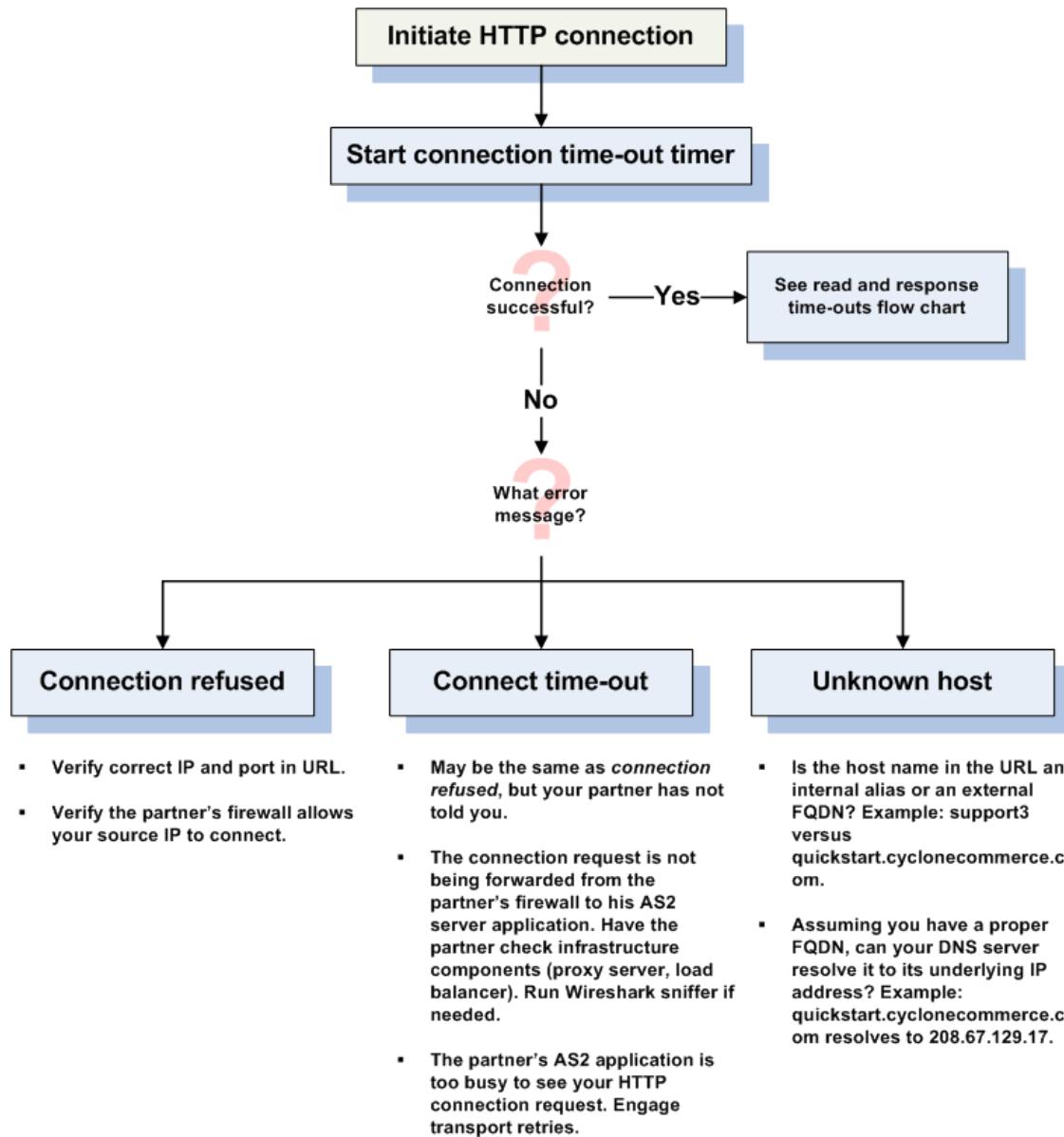


Figure 198. HTTP connection troubleshooting flow chart

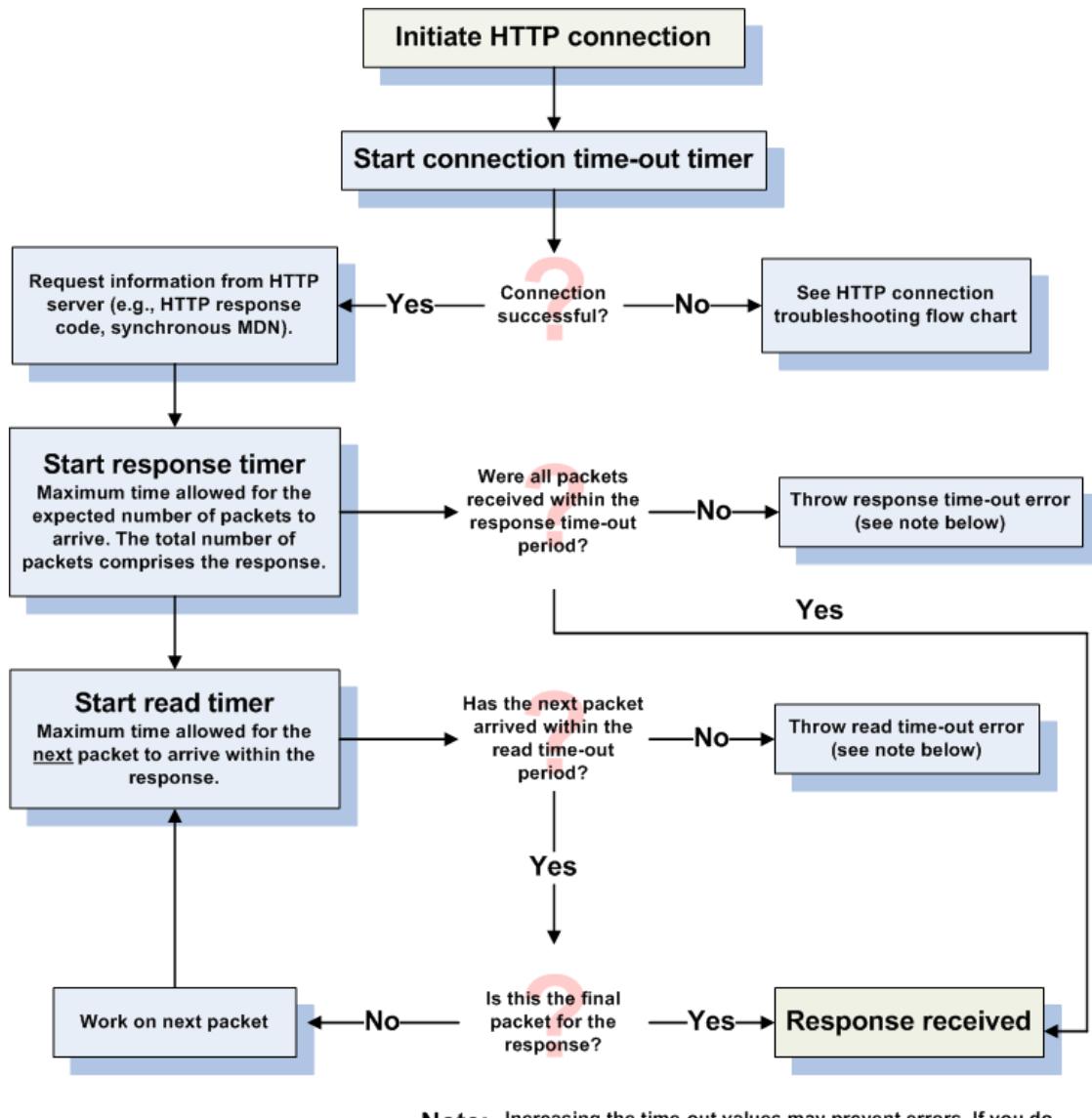


Figure 199. Read and response time-outs flow chart

## Trading large messages

Although Activator has no inherent limitations on the size of the messages it can process, a number of factors — singly or in combination — can affect the capacity for handling large documents.

Activator is designed handle very large messages of 2 gigabytes or more. But external factors can limit the size of messages you can exchange with partners. Such factors can include available disk space and RAM and network hardware, including computers, routers, load balances and firewalls. These factors not only can affect your system, but your partners'.

The following are points to consider if you want to trade very large documents.

## Disk space

Disk space requirements can become very large because the trading engine creates multiple copies of each message.

- 1** The temporary directory, generally located on the local file system, must be large enough to contain multiple copies of each message. See [Temp directory requirement](#) on page 2.
- 2** The backup directory, generally located on a network file system when running in a cluster, must be large enough to hold multiple copies of each message. For example, by default a backup is kept of both the raw “consumed” file and the “packaged” or “unpackaged” file, depending on whether you are sending or receiving. These copies may remain in the backup directory for many days, depending on the purge interval.
- 3** The storage associated with the integration pickup and delivery exchanges must be sufficient to hold one copy of each message that is sent or received.

## Databases, firewalls and large messages

Since the contents of messages are stored on the file system and not in the database, there is no need to allow additional database space to handle large messages. However, if you are trading a large volume of messages, this can be a consideration because a fixed amount of storage is required in the database for each message.

If you have a firewall between the trading engine and the database and you trade large messages that request synchronous AS2 receipts, you must do one of the following to avoid database errors while waiting for synchronous receipts:

- 1** Change your firewall idle time-out to allow connections between the trading engine and the database to remain open and idle for at least as long as it takes for the synchronous receipt to be returned. This

applies to inbound and outbound messages. The amount of time to allocate depends on the size of the largest message and the load on your trading engine, as well as your partners' trading engines. This easily could be hours for multi-gigabyte messages that are signed and encrypted.

or

- 2 Change your collaboration settings to request asynchronous AS2 receipts. You must ask your partners to do this as well.

Also see [AS2 receipts](#) on page 646 for more information.

## ***Considerations by transport***

Certain transports have special considerations for large messages. Check with your vendor regarding considerations for third-party transports such as JMS providers.

The following considerations apply for transport clients and servers embedded in the trading engine.

### **Restarts (FTP and HTTP only)**

If you know your partner's server supports restarts, you may want to enable the attempt restarts check box for the associated delivery exchanges (see [Transport maintenance](#) on page 295). This allows the transfer to continue where it left off if it is interrupted, rather than starting over from the beginning. In the case of FTP you also can enable restarts for pickup exchanges.

### **HTTP chunking**

Sending a message larger than 2 gigabytes can result in problems if chunking is not enabled. For this reason, when the trading engine sends a message larger than 2 gigabytes, it automatically enables chunking for that message, even if chunking is disabled for the partner's HTTP delivery exchange.

Virtually all modern HTTP/1.1-compliant servers support chunking, including the embedded server used by the trading engine. But if a partner's server does not support chunking, and you need to trade message's larger 2 gigabytes, your partner must upgrade his server.

## AS2 receipts

You should request asynchronous receipts from partners if you are trading messages larger than a few megabytes. Failure to do so could result in response time-outs on the client or server or idle time-outs in firewalls or gateways in the connection chain. This is due to the potentially long period in which the connection may appear idle while the server is unpacking the message until it can return the synchronous receipt. Problems related to timed-out connections can be difficult and time-consuming to diagnose. If you know you will be trading even one large message, change your AS2 collaboration settings to request asynchronous receipts before you have problems. See [View or change default settings](#) on page 450.

## AS2 response time-out

Beginning with version 5.4.2, the trading engine automatically adjusts the response time-out for synchronous receipts on both the client and server side to an appropriate value based on the size of the message.

In previous versions the response time-out was fixed at 30 minutes on the server side. On the client side it could be adjusted manually on the Advanced tab for the delivery exchange, but only to a fixed value that was not suitable for both large and small messages.

This change allows trading large messages that request synchronous receipts to be more reliable. Nevertheless, you are strongly encouraged to request asynchronous receipts to avoid potential time outs in external network components that are beyond your control.

## Firewall idle time-out

If you know you will be receiving large messages, adjust your firewall idle time-out accordingly.

For example, if your partners send large AS2 messages that request synchronous receipts (not recommended), you must set your idle time-out to be longer than it takes the trading engine to unpack the largest message and return the response.

Your partner also must make similar adjustments to receive large messages from you.

With multi-gigabyte messages, the time required to return a synchronous receipt could be hours or days, depending on factors such as the speed of the hardware and system load. Even when requesting asynchronous receipts, the message must be copied to the backup directory before the HTTP response can be returned. If the backup directory is on the network,

this could take several minutes. Contact your firewall administrator to ensure the time-out is set appropriately based on the types of receipts being requested and the size of the messages.

## FTP versus HTTP

There is a general belief that FTP is better than HTTP for large messages. If asynchronous receipts are requested and firewall idle time-outs are set appropriately, there is no reason to prefer FTP over HTTP. This is the case particularly when both parties are using Interchange or Activator, which support restarts over HTTP as well as FTP.





# 31 Synchrony integration

The following topics describe how to integrate Synchrony EndPoint Activator 5.5.1 with the following sibling components:

Component	Short name
Synchrony Gateway 6.11.1	Gateway
Synchrony Integrator 3.3	Integrator
Synchrony Sentinel 3.1	Sentinel

Integration is supported with these versions or later of the components.  
Integration with earlier versions is not supported.

## Procedure

- [Integrate with Synchrony Sentinel](#)
- [Integrate with Synchrony Integrator](#) on page 661
- [Integrate with Synchrony Gateway](#) on page 663

## Integrate with Synchrony Sentinel

Use this procedure to integrate Activator with Synchrony Sentinel.

The integration procedure presumes you already have configured Activator to trade e-commerce messages between your community and trading partners.

Copy the file sentinel.zip from [install directory]\[build number]\conf\sentinel. This file contains the tracked object XFBTransfer v3.3. Use Axway Designer to add the tracked object to Sentinel.

You can copy the contents of this ZIP file before installing the trading engine from the following installer directory:

Components\Gateway\_Interchange\extras\sentinel

When integrated with Sentinel, Activator forwards meta-data about events related to e-commerce messages traded between partners, but not payloads.

The following topics describe configuring Activator to integrate with Sentinel, but do not include the Sentinel tasks for integrating with Activator. Users of Sentinel are advised to review the Sentinel documentation.

## Steps to integrate with Sentinel

Use this procedure to configure Activator to integrate with Sentinel

- 1 Select **Trading configuration** on the toolbar to open the community profile page. Click **Configure Sentinel integration** to open the page for integrating with Sentinel.

### Integrate the trading engine with Sentinel

Sentinel

Send to Sentinel

**Save changes**

**Figure 200. Page for integrating with Sentinel**

- 2 Select **Send to Sentinel** to display configuration fields.

### Integrate the trading engine with Sentinel

Sentinel

Send to Sentinel

Primary host: \*

Primary port: \*

Secondary host:

Secondary port:

Enable buffer congestion monitoring

**Save changes**

**Figure 201. Sentinel configuration fields**

- 3 Complete the fields. Click **Save changes** when done

#### Primary host

The fully qualified domain name or IP address of the computer running Sentinel.

If Activator and Sentinel are running on the same computer, you can use the short computer name instead of the fully qualified domain name.

#### **Primary port**

The port Sentinel uses to listen to connections from Activator. The default port is 1305.

#### **Secondary host**

The fully qualified domain name or IP address of the computer running a second Sentinel, if one has been installed and configured.

The secondary Sentinel is used for fail-over. If the trading engine cannot communicate with the primary Sentinel, it tries to connect to the secondary Sentinel.

#### **Secondary port**

The port the secondary Sentinel uses to listen to connections from Activator. The default port is 1305.

#### **Enable buffer congestion monitoring**

Selecting this check box uses the Activator system throttle to make sure events queued for sending to Sentinel will not be lost if the backed-up volume exceeds the specified congestion threshold percentage.

Enabling congestion monitoring is unnecessary unless recommended by technical support.

## ***Event meta-data sent to Sentinel***

The following tables describe the meta-data elements and values Activator can pass to Sentinel. Sentinel recognizes these element names and values. A user of Activator cannot see this information. The trading engine generates the data solely for use by Sentinel. There are two tables:

- ◆ Table 23 lists and describes the meta-data Activator can send to Sentinel for most messages, excluding ebXML.
- ◆ Table 24 lists and describes the additional meta-data Activator can send to Sentinel for ebXML messages.

## Meta-data for most messages

In Table 23 the value column shows, when applicable, an actual or sample value for an element. Sample values are noted as such.

**Table 23 - Meta-data Activator can send to Sentinel**

Element	Description	Value
Application	Name of the sending application.	Interchange
CycleId	When Activator reports its <a href="#">InternalCycleId</a> to Sentinel, Sentinel hashes the value to produce the CycleId.  The sample value is an example of a CycleId as generated by Sentinel. Activator does not generate the CycleId.	Sentinel sample:  XR1QTJr3uCYG71 KWdqJN113RSSx1s X
DeliveryExchange	The name of the trading delivery exchange used to send or receive the message. Naming a delivery exchange is optional when setting up a transport or performing maintenance. If the transport does not have a name, the DeliveryExchange value is blank.	
Direction	The direction of the message related to the event. Outbound is a message picked up from integration for sending to a trading partner. Inbound is a message received from a partner and sent to integration.	O = Outbound  I = Inbound
EventDate	The date, in yyyy-mm-dd format, the trading engine generated the event.	Sample:  2006-11-14
EventId	The event identification number generated by the trading engine.	Sample:  1163521349929
EventTime	The time, in hh:mm:ss:sss format, the trading engine generated the event.	Sample:  14:26:51:268
FileName	The original name of the file the trading engine received from integration or from a partner.	Sample:  pc3_To_pc1_1001.edi

**Table 23 - Meta-data Activator can send to Sentinel**

<b>Element</b>	<b>Description</b>	<b>Value</b>
FileSize	The size of the file in bytes.	Sample: 1636
GmtDiff	The number of minutes between local time and GMT.	Sample: -420
IntegrationDelivery	For an inbound message, the name of the integration delivery exchange used to route a message received from a partner to integration. Naming a delivery exchange is optional when setting up a transport or performing maintenance. If the transport does not have a name, the IntegrationDelivery value is blank.	
IntegrationPickup	For an outbound message, the name of the integration delivery exchange used to retrieve the message from integration. Naming a delivery exchange is optional when setting up a transport or performing maintenance. If the transport does not have a name, the IntegrationPickup value is blank.	
InternalCycleId	The raw unhashed <a href="#">CycleId</a> generated by the trading engine.  This is the key element for linking together the various pieces of a transaction.	Sample: SUIV116352391755 4.1909@tsuliga-t41_te ... E
IsAlert	Indicates whether the transaction is in an alert state.	0 = not alert 1 = alert, not resolved 2 = alert, resolved
IsEnd	Indicates whether the transaction is completed.	0 = transaction not end 1 = transaction completed

**Table 23 - Meta-data Activator can send to Sentinel**

<b>Element</b>	<b>Description</b>	<b>Value</b>
IsException	Indicates whether the transaction is in an exception state.	0 = not exception 1 = exception
IsServer	Indicates whether the trading engine is acting as a server.	0 = not server 1 = server
IsSsl	Indicates when an SSL connection is in use.	0 = not SSL 1 = SSL
Location	Name of the location of the trading engine node sending the event.	Sample: hostname_te
Machine	Name of the computer running the trading engine.	Sample: hostname-t41
Monitor	Indicates the trading engine is the transaction monitor.	INTR = Interchange
MonitorVersion	The version and build number of Activator.	Sample: 5.5.0.0.0.2167
ProductName	Product name of the trading engine.	Interchange
ProductOs	Name of the operating system on the computer running the trading engine.	Sample: Windows XP
Protocol	The transport protocol for the traded messages.	Sample: HTTP
ProtocolId	Name of the EDIINT message protocol.	Sample: AS2
Rappl	URL the trading engine uses to send a message to a partner.	Sample: http://hostname-t41:4085/exchange/partner
ReceiverId	Routing ID of the message receiver.	Sample: partner

**Table 23 - Meta-data Activator can send to Sentinel**

<b>Element</b>	<b>Description</b>	<b>Value</b>
RequestType	The type of message processing request the transaction represents.	S = single transfer G = split documents
RetryMaxNumber	Maximum number of times the trading engine tries to send a message.	Sample: 3
RetryNumber	Number of times the trading engine tried to send a message.	Sample: 1
ReturnCode	Message type.	0 = unknown 1 = request 2 = receipt 3 = request and receipt
ReturnMessage	Description for a rejected transaction.  The reject reasons reported to Sentinel are the same as the reasons used by the trading engine and reported in Message Tracker.	
Sappl	URL a partner uses to send a message to the trading engine.	Sample:  http://hostname-t41:4085/exchange/community
SenderId	Routing ID of the sending community in the trading engine	Sample:  community
Site	Name of the trading engine.	INTR = Interchange
StartDate	Date, in yyyy-mm-dd format, the trading engine originated a message.	Sample:  2006-11-14
StartTime	Time, in hh:mm:ss:sss format, the trading engine originated a message.	Sample:  10:05:17:977

**Table 23 - Meta-data Activator can send to Sentinel**

<b>Element</b>	<b>Description</b>	<b>Value</b>
State	Event state reported to Sentinel by the trading engine.  For a list of states, see <a href="#">Message states reported to Sentinel</a> on page 658.	Sample: To_Ack
TradeDestination	For an outbound message, the routing ID of the receiving partner.  For an inbound message, the routing ID of the receiving community.	
TradeDestinationAlias	For an outbound message, the name of the receiving partner.  For an inbound message, the name of the receiving community.	
TradeFormat	The message protocol used to send the message.	Sample: AS2
TradeMessageId	The ID the trading engine assigned to the message.	Sample: <1179782810657.14 23@a_te>
TradeOriginator	For an outbound message, the routing ID of the sending community.  For an inbound message, the routing ID of the sending partner.	
TradeOriginatorAlias	For an outbound message, the name of the sending community.  For an inbound message, the name of the sending partner.	
TradeSigningAlgo	The algorithm used to sign the message	Sample: SHA1
TradeState	The state of the original message.	Sample: Delivered

**Table 23 - Meta-data Activator can send to Sentinel**

<b>Element</b>	<b>Description</b>	<b>Value</b>
UserObjectId	Cycle ID with message state added as suffix.	Sample: SUIV116352391755 4.1909@hostname _te ... E.To_Ack

## Additional meta-data for ebXML messages

Table 24 shows the additional meta-data the trading engine can send to Sentinel for ebXML messages. These elements are in addition to those in Table 23.

These same elements also are used by Activator. But when Activator passes the meta-data to Sentinel, the trading engine adds the prefix **Ebxml** to the element name.

**Table 24 - ebXML meta-data Activator can send to Sentinel**

<b>Element</b>	<b>Description</b>	<b>Sample value</b>
EbxmlAction	Identifies an action within a Service that processes the message.	Acknowledgment
EbxmlConversationId	A string identifying all consecutive messages belonging to the same collaborative business process instance. This is useful for monitoring and auditing purposes.	oba9577c-5886-4a27-a2b1-d1226cd76155
EbxmlCpaId	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.	urn:Cyclone4-turpinsoft
EbxmlFromRole	The role of the sending party.	Test_E1
EbxmlProcessSpecUUID	Unique identifier of the BPSS ProcessSpecification.	
EbxmlReceiverPartyName	The name of the receiver.	Cyclone4

**Table 24 - ebXML meta-data Activator can send to Sentinel**

<b>Element</b>	<b>Description</b>	<b>Sample value</b>
EbxmReceiverRoutingId	The routing ID of the message receiver.	cyclone4
EbxmReceiverRoutingIdType	The ebXML party ID type of the message receiver.	string
EbxmRefToMessageCoreId	A value for this element is provided when one message refers to another. Typically this would be in web services and ebXML when a business response is sent that refers to a previous message. The value is the core ID of the related message.	ci1181752060047.1 291@tsuliga-t41_te
EbxmSenderPartyName	The name of the sender.	Turpinsoft
EbxmSenderRoutingId	The routing ID of the message sender.	turpinsoft
EbxmSenderRoutingIdType	The ebXML party ID type of the message sender.	string
EbxmService	Identifies the collaborative business process. This can be bound to a service in back-end integration.	urn:oasis:names:tc:ebxml-msg:service
EbxmServiceType	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.	string
EbxmToRole	The role of the receiving party. Indicates the role the party is playing in the business transaction, such as the seller role.	Test_E1

## **Message states reported to Sentinel**

The following are the message states the trading engine can report to Sentinel.

A state marked as optional means the trading engine can report the state only when collaboration settings call for sending or receiving receipts for traded messages.

## Inbound from partner

When the trading engine receives a message from a partner, the following states are reported in order.

- ◆ Created
- ◆ Unpack
- ◆ Receiving
- ◆ Received
- ◆ To\_Ack (optional)
- ◆ Acked (optional)

## Outbound to partner

When the trading engine sends a message to a partner, the following states are reported in order.

- ◆ Created
- ◆ Pack
- ◆ Sending
- ◆ To\_Ack (optional)
- ◆ Sent
- ◆ Acked (optional)

## Outbound to a partner for split document

When the trading engine picks up a message from integration that requires splitting, the following states are reported in order.

- ◆ Created
- ◆ Splitting
- ◆ Split

After splitting, the trading engine reports the following states in order for each split message.

- ◆ Pack
- ◆ Sending
- ◆ To\_Ack (optional)
- ◆ Sent
- ◆ Acked (optional)

## Inbound from partner for re-routed message

When message re-routing has been configured in the [Message handling](#) area of the user interface, the following states are reported in order when the trading engine receives a message from partner A and re-routes the message to partner B.

In Sentinel a re-routed message is reported as two messages linked together. The first message is the one received by the trading engine. The second message represents the re-routing by the trading engine of the first message.

### First (received message)

- ◆ Created
- ◆ Routed
- ◆ Received
- ◆ To\_Ack (optional)
- ◆ Acked (optional)

### Second (re-routed message)

- ◆ Created
- ◆ Pack
- ◆ Sending
- ◆ To\_Ack (optional)
- ◆ Sent
- ◆ Acked (optional)

## Miscellaneous states

In addition to the preceding states, there are some other states that can be reported. These are listed separately, as the states can occur at various times during processing.

### Interrupted

The interrupted state normally indicates something disrupted processing, but the trading engine will try again later. For example, failure to connect to a transport can cause an interrupted state.

### Nacked

The nacked state indicates a message was received, but that some error occurred that requires user intervention to resolve before the message can be resent. For example, an expired certificate can cause a nacked state.

### Rejected

The rejected state indicates processing was stopped for an inbound or outbound message because some error occurred. For example, the trading engine rejects a message when the trading partner is unknown.

## Integrate with Synchrony Integrator

Activator can send messages to and pick up messages from the Synchrony Integrator message translator via a JMS queue.

This topic describes configuring Activator to integrate with Integrator, but does not include the Integrator tasks for integrating with Activator. Users of Integrator are advised to review the Integrator documentation.

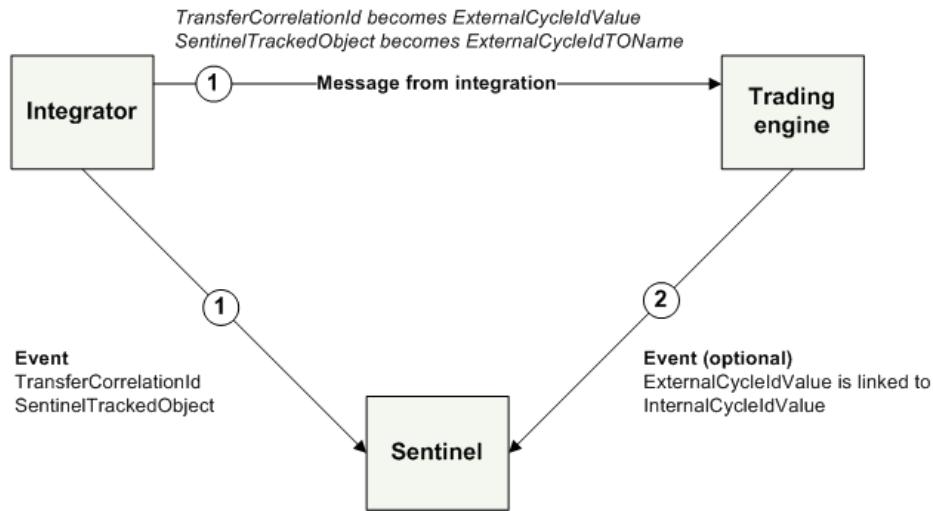
Messages are moved between Activator and Integrator via a special JMS exchange named Synchrony Integrator. Setting up the Synchrony Integrator integration pickup or delivery exchange is exactly like setting up any JMS integration exchange. The only difference is you must select **Synchrony Integrator** as the integration pickup or delivery option in the delivery exchange wizard. Except for that, configuring the transport is just as described in [JMS transport](#) on page 245. Integrator must be configured to use the same JMS queue as Activator.

**Note:** The Synchrony Integrator exchange only is for performing JMS integration with Integrator. If you do not want to integrate with Integrator, use a normal JMS exchange.

Adding a Synchrony Integrator exchange for routing messages from partners to integration creates a special event router. The router moves meta-data between Activator and Integrator. The meta-data values allow each component to link messages exchanged between them. If Activator also is configured to integrate with Synchrony Sentinel, the meta-data correlates the messages among all three components.

### ***Message picked up from Integrator***

Figure 202 illustrates the flow when Activator picks up a message from Integrator.



**Figure 202. Activator picks up message from Integrator**

The following steps describe the flow.

- 1 The trading engine picks up a message from integration. Attached to the message as a string property are Integrator meta-data elements. The trading engine changes the element names.

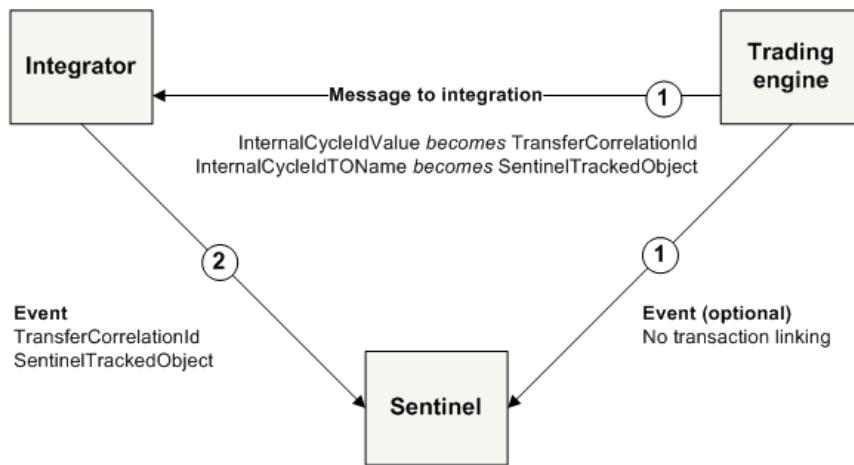
Integrator meta-data	The trading engine changes to
TransferCorrelationId	ExternalCycleIdValue
SentinelTrackedObject	ExternalCycleIdTOName

At the same time, Integrator can send events to Sentinel.

- 2 The trading engine processes the message. If Sentinel integration is enabled, the trading engine sends events, but also sends information to link ExternalCycleIdValue to its newly generated InternalCycleIdValue. Sentinel can use the information to link to the information received from Integrator in step 1.

## Message sent to Integrator

Figure 203 illustrates the flow when Activator sends a message to Integrator.



**Figure 203. Activator sends message to Integrator**

The following steps describe the flow.

- 1 The trading engine sends a message received from a partner to integration. Before doing so, the trading engine changes names of key meta-data to Integrator-friendly names.

The trading engine changes	To Integrator-friendly meta-data
InternalCycleIdValue	TransferCorrelationId
InternalCycleIdTOName	SentinelTrackedObject

If Sentinel integration is enabled, the trading engine sends events to Sentinel, but no transaction linking information is included.

- 2 Integrator processes the message received from the trading engine. Integrator also can send events to Sentinel. The meta-data Integrator sends links back to the meta-data received from the trading engine.

## Integrate with Synchrony Gateway

Use this integration to have a community in Activator poll and retrieve files from Synchrony Gateway. Clear-text files are retrieved via HTTP. Upon receipt, Activator delivers the files to back-end integration, just as it does for any message received from a partner.

From the Activator perspective, this integration is set up almost like a typical trading relationship between a community and partners who exchange e-commerce business messages. But in this case, the trading engine is not exchanging messages, only retrieving files from a Synchrony Gateway queue. The retrieved files are not packaged, encrypted or

compressed. The trading engine sends the retrieved files to back-end integration without further processing. Activator does not send Synchrony Gateway receipts or any other messages.

Once Activator has retrieved a file, Synchrony Gateway marks its copy of the file as processed successfully.

## **Synchrony Gateway configuration**

Use this procedure to enable Synchrony Gateway to integrate with Activator.

### **Steps**

- 1** Set up HTTP virtual file directories or real file directories for users who represent the instances of Activator. One VFD or RFD is required for each user. Activator polls the VFD for files to retrieve. See the Synchrony Gateway help for how to add VFDs and RFDs.
- 2** Edit the template.html file to include only the following lines:

---

```
<XFER_LIST>
  <REAL_FILE>
    <FILE_INFO>
      <FNAME>@fname@</FNAME>
      <SIZE>@SIZE@</SIZE>
      <DATE>@DATE@</DATE>
    </FILE_INFO>
  </REAL_FILE>
  <XFER_AVAIL>
    <FILE_INFO>
      <FNAME>@FNAME@?local_ident=@LOCAL_ID@%&%I</FNAME>
      <SIZE>@SIZE@</SIZE>
      <DATE>@DATE@</DATE>
    </FILE_INFO>
  </XFER_AVAIL>
</XFER_LIST>
```

---

Template.html is at [install directory]\samples\web\templates.

Including this code directs Synchrony Gateway to generate a list of files available to consume. Each listed file is separated by a line feed.

The template.html file is the default template. In a production environment the default template is left in place. A special template file, edited as described here, is created. The Synchrony Gateway administrator assigns the special template to the users of Activator who are retrieving files from Synchrony Gateway.

- 3 For Activator to connect with a user name and password, web authentication must be set instead of CGI (common gateway interface) in Synchrony Gateway.

Select **Session > Local** and right-click and select **Configure** to open the configuration dialog window. Select **Connectivity > Internet protocols > HTTP** and click **Options**. Under server identification, select **Web** in the Method field drop-down list.

- 4 If Activator is to connect via HTTPS (HTTP over SSL) rather than HTTP, run the following sample scripts in this order:

- a cert.bat
- b sprof.bat
- c netprof.bat

The Synchrony Gateway server must be running when you execute the scripts. The scripts are at [install directory]\samples\tls\case1.

The scripts import sample certificates and create two security profiles and many network profiles.

Add a port specification, using **6331** as the port and **SPROF\_IN** as the Transport Layer Security (TLS) profile. The netprof.bat script creates port 6331; the sprof.bat script creates SPROF\_IN.

Restart the Synchrony Gateway server for the changes to take effect.

**Note:** If the security options are disabled when creating the port specification, return to the configuration dialog window and change the SecureRelay access policy field to **Through SecureRelay and directly**. You must delete all existing port specifications to do this.

- 5 Changing settings for dynamic server processes is recommended.

Select **Session > Local** and right-click and select **Configure** to open the configuration dialog window. Select **Connectivity > Internet protocols > HTTP** and click **Options** and then click **Advanced**. Under dynamic server processes set these values:

No. processes min 15  
No. processes max 30  
Balance ratio 25

- 6 Provide information to the Activator administrator. This can include:

- The URL for connecting to Synchrony Gateway. Each URL is unique per partner. The following are examples of URLs.

- HTTP: http://host:port/user1/transfer/partner1/
- HTTPS: https://host:port/user2/transfer/partner1/

Where:

- host is the fully qualified domain name or IP address of the computer running Synchrony Gateway
- port is the number of the port Synchrony Gateway listens for connections from Activator
- If HTTPS, provide a file containing a certificate and public key. Activator must import and trust this certificate.
- The user name and password Activator must use to connect.

## ***Activator configuration***

Use this procedure to enable Activator to integrate with Synchrony Gateway.

### **Steps**

- 1 Add and configure a community profile. See [Add a community](#) on page 249.

This can be the same profile as the one used to exchange e-commerce messages with partners other than Synchrony Gateway. If you use the same profile, make sure the delivery exchange added in step 3 is not the default. The default should be the exchange you want partners to use for sending e-commerce message to your community.

- 2 Add a partner profile to represent Synchrony Gateway. See [Add a partner](#) on page 254.

Adding this partner is not required, but doing so makes it easier to use [Message Tracker](#) to search for messages retrieved from Synchrony Gateway. Use a partner name you can identify with Synchrony Gateway.

When adding the profile, only a partner name and contact name are needed. Associate the profile with your community. No other setup is required, including no delivery exchange or certificate.

- 3** On the community profile summary page, click **Delivery exchange** in the navigation graphic at the top of the page. On the Pick a delivery exchange page, click **Add a delivery exchange** to open the delivery exchange wizard.
- 4** Select **Other** and click **Next**.
- 5** For the From address, select **Specify the address. Always use a fixed address**. Click **Pick party** and select the Synchrony Gateway partner as the sender. Click **Next**.  
  
If you did not add a partner profile in step 2, select your community as the sender.
- 6** For the To address, select **Specify the address. Always use a fixed address**. Click **Pick party** and select your community as the receiver. Click **Next**.
- 7** Select **HTTP** and click **Next**. The Synchrony Gateway HTTP interface delivery exchange is selected by default. Click **Next** to open the HTTP configuration page.



**Figure 204. HTTP configuration page**

- 8** Complete the fields and click **Finish**.

### 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 not encrypted.

***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 to connect to the server.

- 9** If using SSL, import to your community the certificate and public key being used by Synchrony Gateway.



---

# Index

## A

access.log 96  
acknowledgments 505  
admin user 78  
AIX LVM 18  
alerts.xml  
    editing the file 123  
    e-mailing alerts 121  
as1Tool 45  
as2Tool 45  
as3Tool 45  
asxTool 45  
asynchronous acknowledgment 455  
auto import of root, intermediate certificates 395  
automounting on UNIX 17

## B

build number directory description 50

## C

CEM  
    overview  
    request message 429  
    response message 429  
    send request 431  
    track requests 432  
certificate exchange messaging  
    see CEM  
certificate revocation lists (CRLs) 437  
certificates  
    basic information 388  
    CRLs 437  
    deleting 427  
    dual keys 385  
    dual-key 382  
    Entrust 408, 411, 412  
    expiration 392  
    exporting yours to a file 424  
    list of, displaying 397  
    matching fingerprints 405  
    new, generating or loading 407

partners with interoperable software 391  
PKI 380  
revocation 382  
RSA Keon 408, 415  
SSL authentication 388  
troubleshooting 445  
viewing 401  
when to obtain new 391  
certScan 45, 445  
collaboration settings  
    AS1 defaults 451  
    AS2 defaults 454  
    AS3 defaults 459  
    community 474  
    default 450  
    encrypt messages 452, 457, 461, 464, 467, 469, 476, 480  
    overview 447  
    partner 477  
    reliable messaging 473  
    RosettaNet 1.1 defaults 467, 468  
    secure e-mail defaults 466  
    secure file defaults 463  
    view 448  
    web services defaults 470  
command set document, for FTP client 523  
common directory description 51  
companies vs. communities 65  
components of the application 148  
compress messages option 452, 456, 460, 464  
compressing messages 452, 456, 460, 464  
configuration  
    for test trading 529  
configuration tool for database 52  
control ID duplicates 481  
core.log 96  
CRL  
    distribution point 439  
    using 437  
CSOS  
    CRLs 612

---

- events 138
- order signing 625
- overview 607
- purchase orders page 612
- receive configuration 610
- send configuration 609
- signing certificate 611

## D

- data backup, purging 515
- database
  - configuration tool 52
  - Derby JDBC driver port 9
  - large messages 644
- dataMover 45
- db\_configurator.log 96
- db2\_runstats 46
- dbConfig 44
- dbConfig database tool 52
- default ports 9
- delete records, backups 520
- DeleteUploadedDirectorDocs.sql 46
- deleting certificates 427
- delivery exchanges
  - directing inbound files 273
  - enable 274
  - four required 202
  - integration 206
  - naming 212
  - post-processing 265
  - set preferences 276
  - test 274
  - trading 204
  - wizard 212
- Derby
  - JDBC driver port 9
- detached e-mail for community 213
- diagnose 46
- dirTester 46
- disaster recovery planning 16, 56
- docgen command (UNIX) 535
- Document Generator
  - using 535
- documents
  - generating test 535
- duplicate CSOS orders 485, 624

- duplicate EDI messages 481
- duplicate message handling 481

## E

- ebXML
  - HL7 576
  - message meta-data document 551
  - overview 541
  - ping a partner 551
  - resources on Internet 541
  - STAR BODs 574
  - trading via intermediary 560
- ebxmlCpaSchematronValidator 46, 570
- ebxmlCpaSecurityGuard 46, 571
- ebxmlCpaValidator 47, 572
- EDI
  - test documents 535
- EDI splitter 370
- edit
  - CRLs 437
- e-mail transport for partners 216
- embedded e-mail for community 215
- embedded FTP 230
  - community server fields 190
  - configuration 230
  - hosted partner permission 231
  - maintenance 325
  - programmatically submit 330
- embedded HTTP for community 219
- embedded HTTP server
  - about 183
  - change 375
  - change community 187
  - community fields 188
  - global 184
  - port 10
  - use cases 194
- embedded servers
  - about 183
  - community level 183
  - global HTTP 184
  - global SMTP 185
  - global web services API 186
- embedded SMTP server
  - about 183
  - change 375

- 
- change community 187
  - community fields 192
  - global 185
    - port 10
  - embedded web services API server
    - global 186
  - encryption
    - hybrid strategy 384
    - messages, electing 452, 457, 461, 464, 467, 469, 476, 480
  - encryption keys
    - length, selecting 409
    - public/private 384
  - Entrust certificates 408, 411, 412
  - environment command 44
  - error.log 96
  - events
    - complex filters 111
    - configuring 106
    - delete from database 521
    - event filters 109
    - event routers 107
    - JMS event router 114
    - levels 105, 127
    - list 126
    - log file event router 112
    - MetadataProcessors 108
    - Oracle AQ event router 114
    - overview 105
    - simple filters 110
    - tables 126
  - events.events.xml file 106
  - executive log 95
  - export certificate to file 424
  - export community profile 167
  - exportProfile 47
  - external SMTP server 23
- F**
- file system transport 223
  - fillInMessageDirection 47
  - filtering events 109
  - fingerprints in certificates 405
  - firewalls
    - embedded FTP 235
    - idle time-out 646
- large messages 644
  - working with 171
- FTP**
- scripting 523
  - transport configuration 226
  - troubleshoot 226
- FTP client**
- command set document 523
  - command set document, editing 524
  - meta-commands 524
- FTP embedded server**
- see embedded FTP
- ftpTester 47, 525
- G**
- generate cluster thread dumps 90
  - global embedded HTTP server 184
  - global embedded SMTP server 185
  - global embedded web services API server 186
  - global HTTP
    - proxy 377
  - global user settings 85
- H**
- handler
    - WS-Addressing 260
    - WS-Security 260
  - hardware requirements 1
  - help
    - for users of earlier versions 66
    - technical support 12
  - help sysitm display troubleshooting 7
  - HL7 576
  - home page 89
  - host\_environment 43
- HTTP**
- 102 processing 311
  - connection troubleshooting 641
  - embedded servers 183
  - error codes 306, 359
  - proxy 377
  - server logs 96
  - transport for partners 221
  - transport options 637
  - UI port 10

---

**HTTPS**  
    for UI 58  
    UI port 10  
**httpTester** 47

**I**

**import**  
    CA certificate with password 419  
    third-party certificate 418  
    trusted roots 395

import partner profile 168

inbound message page 481

inbound message validation 481

installation directory  
    description of build number directory 50  
    description of common directory 51  
    description of subdirectories 49

installing  
    outline of tasks 15  
    pre-install tasks 17

installing on UNIX  
    automounting 17

InstallLog.log 96

integration delivery exchanges 206

integration directories 273

intermediary ebXML 560

interoperability 153, 206

**J**

JMS event router 114

JMS transport 245

jmsTester 47, 252

JRE directory 49

**K**

KeyInfo element for CPA 565

keyInfoWriter 47

keys  
    length, selecting 409  
    public/private (asymmetric) 384

**L**

large messages 643

levels of events 105, 127

license.xml file  
    about 12  
    replacing 12

list of certificates 397

listTimeZones 47

log on procedure 22

log on the first time 22

log4j.properties 94, 98

Logical Volume Manager 18

logs  
    event router 112  
    event.xml defaults 106  
    file types 94  
    HTTP server 96  
    system 94, 98  
    user interface 96  
    using events.xml 106  
    viewing 96  
    Windows tailing 97

logViewer 48

LVM 18

**M**

manageNode 44

MDNs 505

message actions  
    define actions 492  
    define attributes 491  
    description 487  
    operator definitions 490  
    overview 487

message meta-data document  
    see MMD

message packaging 447

message protocols 204

message receipts 505

message tracker  
    about 92  
    delete search results 506  
    receipts 505  
    reprocess search results 506  
    resend search results 506  
    searching 495  
    using 495  
    view details 503

message validation

- 
- about 481
  - duplicate CSOS orders 485, 624
  - duplicate messages 481
  - messagePurgeTool 48, 520
  - messages
    - compression 452, 456, 460, 464
  - meta-commands for FTP client 524
  - MMD
    - ebXML 551
    - RosettaNet 601
    - web services message protocol 261
  - mmdGenerator 48, 573
  - monitor system 89
  - MQSeries transport 253
- N**
- navigation graphics 65
  - netConfig 44
  - netInfo 48, 54
  - NFS best practices 17
  - nodeInfo 44
- O**
- Oracle AQ event router 114
  - outbound collaboration settings 447
- P**
- partner profile
    - add 164
    - data form 178
    - export and import 166
    - search for 165
  - partyInfo 48
  - password
    - for importing CA certificate 419
  - pedigree
    - events 139
    - log4j troubleshooting 100
  - PKCS#12 file
    - loading a new certificate 419
  - PKI overview 380
  - ports
    - Derby JDBC driver 9
    - embedded HTTP, HTTPS 10
    - embedded SMTP 10
    - possible conflicts 9
- R**
- ready subdirectory 224
  - receipts 505
    - Received-Content-MIC values 506
  - rejectInprocessMessages 48
  - RFC 959 523
  - RosettaNet
    - add DTD-based PIP 588
    - add schema-based PIP 588
    - asynchronous responses 590
    - duplicate checking 586
    - message meta-data document 601
    - overview 585
    - synchronous responses 590
  - routers for events 107
  - routing IDs 165
  - RSA Keon certificates 408, 415

---

## S

searching with wildcards 165  
secondary IDs 42  
security considerations 11  
security glossary 379  
security overview  
    algorithms and key lengths 384  
    certificate maintenance 391  
    encryption/decryption steps 385  
    exchanging company profiles and certificates 390  
    reasons for encryption and certificates 383  
security settings  
    sign documents 453, 457, 461, 464, 467, 468, 469, 477, 480  
self-signed certificates  
    creating 407  
server  
    stop 24  
SFTP  
    transport configuration 240  
sftpTester 48  
signing documents  
    electing 453, 457, 461, 464, 467, 468, 469, 477, 480  
single sign-on 54, 86  
sitemap.log 96  
SMTP embedded servers 183  
SMTP server for system 23  
software requirements 3  
split EDI documents 370  
SSL authentication 388  
SSO 54, 86  
staged HTTP  
    deploying 279  
    file forwarding 291  
    files to deploy 279  
    managing mailboxes 284  
    overview 277  
    servlet log on 283  
    transport configuration 288  
STAR BODs 574  
startServer 44  
statistics monitor  
    overview 90  
    port 9

stop server 24  
stopServer 44  
support, contacting 12  
supported transports 211  
sysInfo 48  
system  
    default ports 9  
    external SMTP server 23  
    hardware requirements 1  
    HTTP proxy 377  
    license.xml file 12  
    log files 94  
    log on procedure 22  
    logging on the first time 22  
    monitor activity 89  
    monitor nodes 90  
    outline of install tasks 15  
    pre-install tasks 17  
    security considerations 11  
    software requirements 3  
    technical support 12  
    uninstalling 24  
    Windows service 19  
system logs 94, 98  
system management 90  
systemuser 44

## T

tail logs in Windows 97  
tail.exe for logs 96  
technical support  
    cluster thread dump 90  
    contacting 12  
    send log files 46, 103  
temp directory  
    change 2  
    requirements 2  
testing  
    completing 533  
    e-mail troubleshooting 532  
    preparing to test trade 529  
    preparing to trade 531  
toolbar 63  
trading delivery exchanges 204  
trading engine  
    about message tracker 92

- 
- data backup, purging 515
  - getting started 147
  - interoperability 153, 206
  - maintenance tips 154
  - major components 148
  - security guidelines 153
  - trading large messages 643
  - trading test documents 531
  - transports
    - detached e-mail for community 213
    - e-mail for partners 216
    - embedded e-mail for community 215
    - embedded FTP 230
    - embedded HTTP for community 219
    - file system 223
    - FTP 226
    - HTTP for partners 221
    - JMS 245
    - JMS integration semantics 246
    - MQSeries 253
    - naming 212
    - post-processing 265
    - SFTP 240
    - supported 211
    - wizard 212
  - trusted roots
    - importing 395
- U**
- UI with HTTPS 58
  - ui.log 96
  - uninstalling system 24
  - UNIX
    - automounting 17
  - unlockuser 45
  - update 49
  - upgrade planning 56
  - upgrade strategy 25
  - upgradeCompare 49
  - upgradeDiff 49
  - upgradeList 49
  - use cases for embedded HTTP server 194
  - user documentation comments xiii
  - user interface
    - about message tracker 92
    - admin user 78
- home page 89
  - HTTP port 10
  - HTTPS port 10
  - logs 96
  - monitor activity 89
  - navigating 63
  - navigation graphics 65
  - proxy servers 73
  - toolbar 63
  - users and roles 77
- user license file 12
- users
  - admin 78
  - change passwords 79
  - global settings 85
  - manage 78
  - manage roles 79
- using CRLs 437
- V**
- view logs 96
  - viewing
    - certificates 401
- W**
- webMethods SSL partner 389
  - wildcard searches 165
  - Windows service 19
  - wizard for delivery exchanges 212
  - working subdirectory 224
- X**
- XML
    - test documents 535
  - XML for JMS, AQ event routers 114

