ECXpert Support for AS2 Messaging

Sun™ ONE Integration Server B2B

Version 3.6.2, Update 1

817-3723-10
July 2003

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.

Contents

List of Figures

List of Tables

List of Procedures

Chapter 1

Getting Started with AS2 Messaging: Installation and Configuration

This guide describes Sun™ Open Net Environment (Sun ONE) Integration Server B2B Edition 3.6.2 (ECXpert) support for Applicability Statement 2 (AS2). It provides the information you need to understand, install, and configure ECXpert for AS2 messaging. It covers the following topics:

- Overview – provides a context for understanding AS2 messaging with ECXpert.
- Setting Up the AS2 Plug-in – gives procedures for downloading, installing, and using the AS2 plug-in with ECXpert.
- Starting and Stopping the Servers – describes the proper order for starting and stopping all the servers in the AS2 system (ECXpert, AS2 Servers, and the Web Server).
- Uninstalling AS2 – provides instructions for restoring the system to its state prior to installing the AS2 plug in.
- What to Do Next – gives suggestions for what to do after installing the plug-in.

This document assumes that you are familiar with basic reliability and security concepts and that you have some understanding of the transport protocols involved.

Overview

Large-scale retailers are increasingly implementing AS2 trading solutions to eliminate their need for *value-added networks* (VAN). A VAN is a private network provider that companies hire for monthly and per-character fees to transport their business documents to and from trading partners. With the ubiquitous use of the World Wide Web, the internet is often a more cost-efficient way for companies to move their data. However, there have been security, reliability, and non-repudiation challenges. AS2 is fast becoming the business document transport protocol for large retailers because it meets these challenges by offering:

- Transport binding for business documents
- Synchronous and asynchronous connection models for Message Disposition Notifications (MDNs)
- Security using S/MIME v3—message integrity, identity, and confidentiality
- Reliable delivery
- Performance benefits through data compression

The AS2 specification describes how to exchange business data securely and reliably using HTTP as an underlying transport. The data is packaged using standard MIME content types so you can use XML, EDI, binary data, and any other data describable in MIME. Message security (authentication, confidentiality) is implemented using S/MIME. Message reliability is enabled through the use of MDNs. Nonrepudiation and Nonrepudiation of Receipt are business/legal concepts that build upon the security and reliability components in AS2.

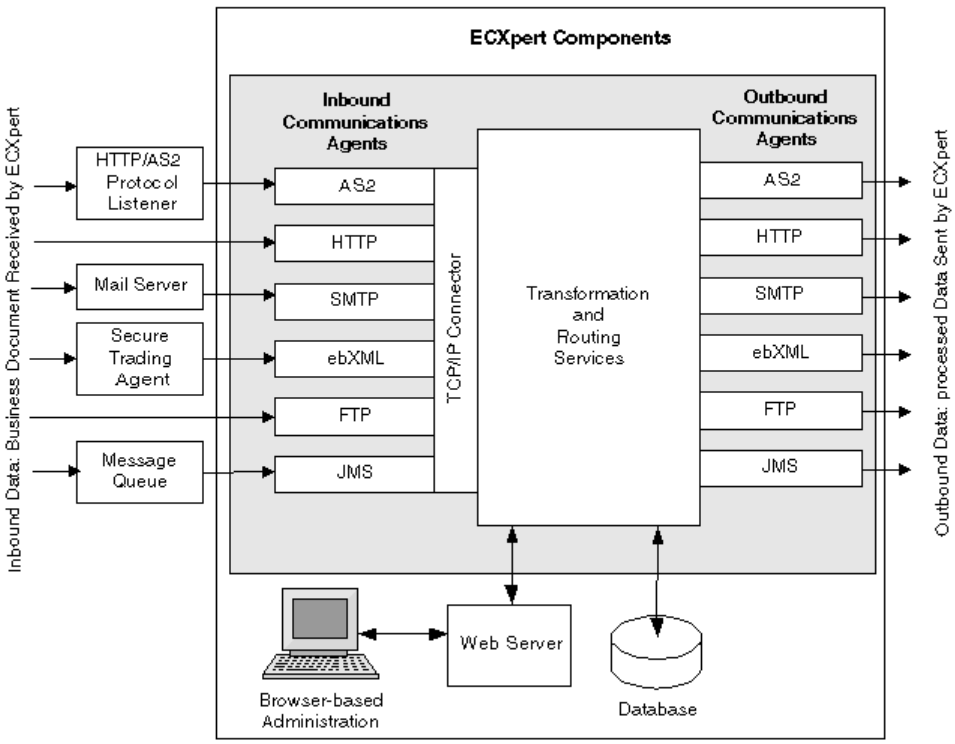
This section provides background for understanding AS2's role with respect to Sun ONE Integration Server B2B Edition 3.6.2 (*ECXpert*) and describes high-level architecture and process flow for the Sun ONE AS2 implementation.

ECXpert and Its Communications Agents

Trading partner applications send data files via ECXpert using their native messaging protocols, for example, FTP, HTTP, JMS, ebXML, SMTP, or AS2. ECXpert itself does not interpret all the different protocols. It uses inbound and outbound Communications Agent components as interpreters as illustrated in Figure 1-1.

On the inbound side, the appropriate Communications Agent accepts the message, extracts the data within it, and submits the data to ECXpert for processing. After processing the submitted data, ECXpert passes the data and messaging parameters to the outbound Communications Agent. The outbound Communications Agent constructs a message in the protocol appropriate for the receiving application using the messaging parameters, and then transmits the message according to the rules of that particular protocol.

Figure 1-1 ECXpert's Messaging System



As with all of the Communications Agents, the AS2 feature provides two agents: an inbound, or receive, Communications Agent and an outbound, or send, Communications Agent. The rest of this section describes these Communications Agents.

Receiving and Sending AS2 Messages

AS2 receive processing depicted in Figure 1-1 starts when the HTTP/AS2 Protocol Listener (or *Protocol Listener*) receives an AS2 message via HTTP POST from the remote trading partner. The receive processing ends when the AS2 Inbound Communications Agent submits the message payload to ECXpert core processing for parsing, translation, and routing as required.

The Protocol Listener is implemented as a servlet. Its job is to process inbound HTTP POST requests sent by the trading partner application. The AS2 Inbound Communications Agent handles all the details of the AS2 protocol processing. It authenticates the message, decrypts it if it is encrypted, and performs MDN generation and reconciliation if required before passing the business data to ECXpert.

The send processing starts as a response to an event, such as submitted data or a scheduled event. Referring to Figure 1-1, after ECXpert completes processing business data to transmit using AS2, it passes the data to the AS2 Outbound Server (a component of the AS2 Outbound Communications Agent). Based on outbound AS2 partnership parameters, the Outbound Server constructs a valid AS2 message with the appropriate type and MDN request attributes. The payload is encapsulated inside an AS2 message envelope. The message is optionally signed, encrypted, or signed and encrypted. The Sending Agent sends the message to trading partner.

Throughout its operation, the Outbound Server uses its logging interface to record events to the ECXpert Event Log, which you can access through ECXpert's Tracking page under the Event Log tab.

Setting Up the AS2 Plug-in

As the AS2 administrator, you perform the following tasks to install the AS2 Communications Agent:

1. Configure your Sun ONE Web Server for use with the AS2 plug-in by creating and configuring a Sun ONE web server instance.
2. Run the install program. This installs the AS2 Inbound and Outbound Communications Agents and configures ECXpert to work with them.

This section provides the details for performing these tasks.

Set the following environment variables in the shell in which you will perform the installation:

- `IWS_SERVER_HOME`

  The root directory of your Sun ONE Web Server installation. (For example, `/export/home/ecxadmin/iWS60SP5`)

- `BDGHOME`

  The root directory of your Sun ONE Integration Server B2B Edition installation. For example, set `$BDGHOME` to `/export/home/actraadm/ecx36/NS-apps/ECXpert` if you have accepted the default location.

Create the HTTP/AS2 Protocol Listener

Ensure that the following products are installed and running:

- Oracle® 8i
- The Administration Server for the Sun ONE Web Server 6.0 SP5

NOTE ECXpert should be shut down for this procedure.

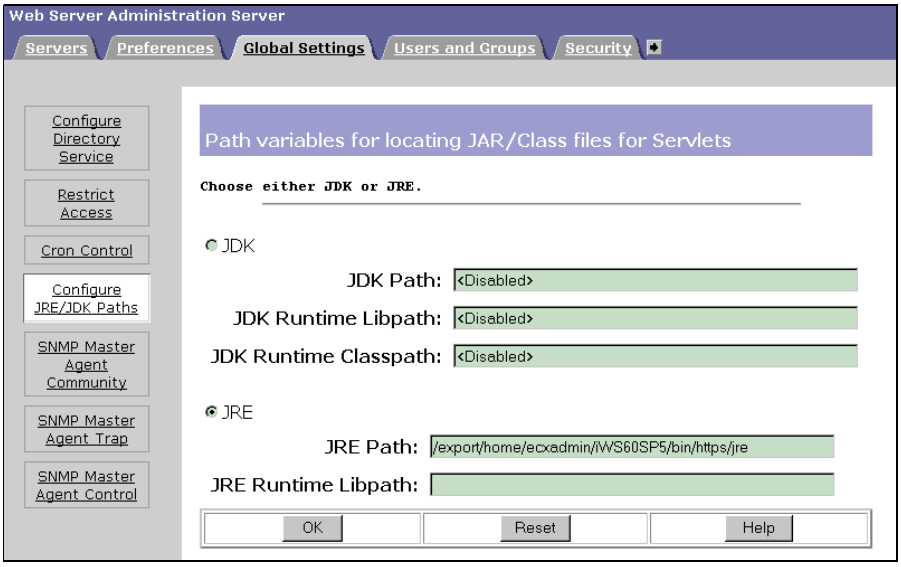
Configure the Sun ONE Web Server to use JRE 1.4. For the example shown in these procedures, `IWS_SERVER_HOME` represents `/export/home/ecxadmin/iWS60SP5/`.

Figure 1-2 illustrates the procedure.

➤ **To configure the Web Server to use JRE 1.4**

1. Open the Web Server Administration Server interface from a web browser.
2. Click the Global Settings tab.
3. Click the Configure JRE/JDK Paths link in the list of links on the left side of the page. The resulting screen should be similar to Figure 1-2.

Figure 1-2 Configure JRE/JDK Paths Screen



4. Update the JRE Path to point to your B2B 1.4 JRE.
5. Click Ok and restart the web server as directed.
6. Close the browser window.
7. Restart the Web Server Administration Server using the `stop` and `start` commands in `$IWS_SERVER_HOME/https-admserv`. Table 1-1 lists an interactive session that stops and starts the sever.

Table 1-1 Example Listing for Starting and Stopping the Web Server

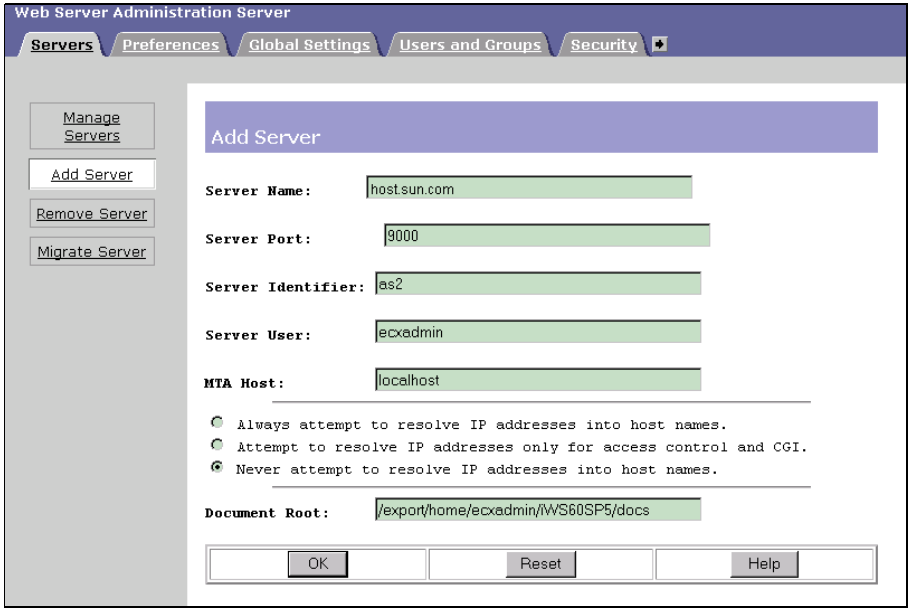
```
ecxadmin@host[85]% ./stop
shutdown: server shut down
ecxadmin@host[86]% ./start
iPlanet-WebServer-Enterprise/6.0SP5 B10/31/2002 16:22
[LS ls1] http://host.sun.com, port 5770 ready to accept
requests
startup: server started successfully
```

➤ **To create the HTTP/AS2 Protocol Listener**

1. Reopen the iPlanet Web Server Administration Server interface.
2. Click the Add Server link on the left side of the Servers tab page.

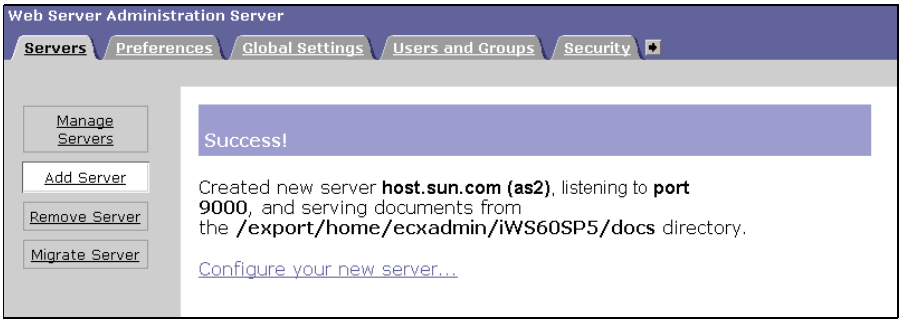
   See Figure 1-3 for an example of the screen.

Figure 1-3 Add Server Screen



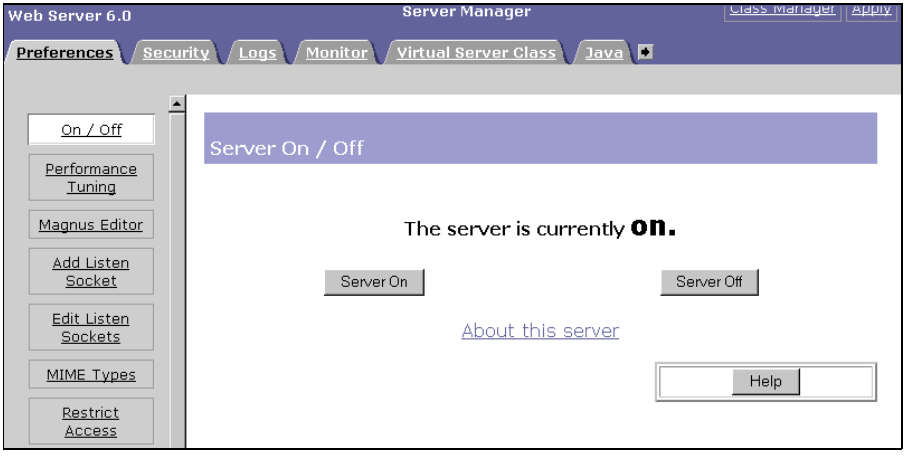
3. The fully qualified server name appears by default in the Server Name field.
4. Set the server port value to an unused port (9000 in the example).
5. Set the server identifier to the recommended default, as2.
6. Set the server user to the ECXpert login name (excadmin in the example).
7. Accept the default MTA Host, localhost.
8. Click the OK button. Figure 1-4 shows an example of the resulting screen.

Figure 1-4 Add Server Success Screen



9. Click the Configure your new server... link.
10. Click the Server On button. This results in the screen shown in Figure 1-5.

Figure 1-5 Server On Screen



➤ To Change https-as2 Ownership

1. Change the file system ownership of the `$IWS_SERVER_HOME/https-as2` directory contents to be the same as the owner of the server instance.

   For example, the administrator in the example would change the owner to `ecxadmin`. If the file ownership is left as root, it could cause problems during deployment.

Install the AS2 Plug-In

These instructions assume that you have already downloaded the AS2 plug-in from the Sun website.

Preconditions for Installation

Ensure that the environment variables `IWS_SERVER_HOME` and `BDGHOME` are set as described on page 15.

You need the following access information:

Table 1-2 Access Checklist

Required Information	Description	Default
ECXpert administrator login and password	Administrative ID and password required to configure ECXpert services and access membership/partnership information.	ECX, ECX
Oracle Hostname	The hostname of the machine on which the Oracle database is installed.	[*hostname*]
Oracle Port Number	The port number of the Oracle listener.	1521
Oracle SID	The SID for the ECXpert database.	ECX
Web Server Instance Name	The name of the web server instance you will create.	as2

➤ **To set up the new files**

1. Uncompress and untar (on Solaris™) the installation software from the file that was downloaded into a temporary directory.

➤ **To install the AS2 plug-in components**

2. Execute the `setup.sh` script located in the root of your temporary directory.

   You should see a listing similar to that shown in Table 1-3.

Table 1-3 Example Listing for AS2 Installation

```
ecxadmin@host[94]% ./setup.sh
## AS2 Plugin Installer ##
##########################

- B2B Configuration Settings -
```

Table 1-3 Example Listing for AS2 Installation *(Continued)*

```
B2B Admin User [ECX]:
B2B Admin Pass [ECX]:

- Oracle Configuration Settings -
Oracle Host [host]:
Oracle Port [1521]:
Oracle SID [ECX]: HOST
Deploying AS2 WAR file to iWS ...

AS2 Server Instance Name [as2]:
[wdeploy] The war file name is
/export/home/ecxadmin/ecx36/NS-apps/ECXpert/http_as2/as2.war
[wdeploy] Deploying web application
[wdeploy] Loading new configuration
[wdeploy] Web application deploy successful
```

3. Fill in the required information accepting defaults where appropriate up to the AS2 Server Instance Name.

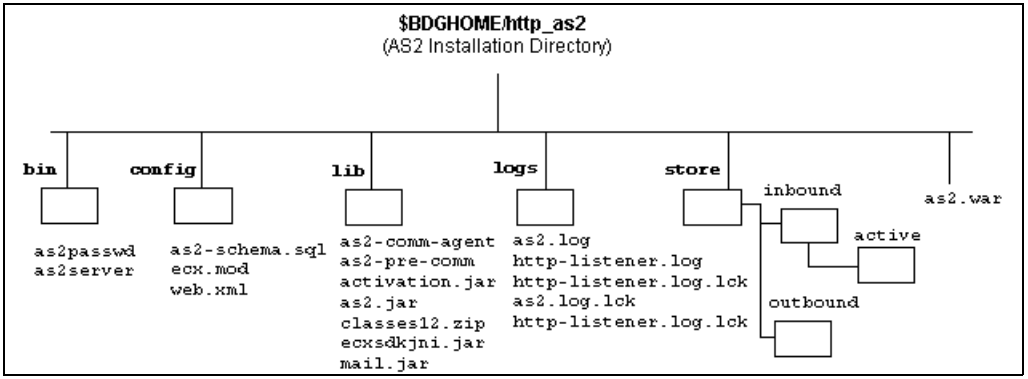
   If you have the admin account with login and password as ECX, ECX; and Oracle is installed on the same machine, you should be able to accept all of the defaults. In the example shown above, the administrator had an Oracle SID different from the default.

4. Fill in the server instance name. If you created the server as https-as2, then accept the default.

5. Watch the status messages returned by the installation program. You know that the installation is a success when you see the message `Web application deploy successful`.

   The resulting directory structure is shown in Figure 1-6.

Figure 1-6 AS2 Directory Structure



The installation program installs all the necessary AS2 components in the correct directory structure locations. It configures the `ecx.ini` file so that the AS2 Communications Agents appear as choices in ECXpert's Partnership Protocols Administration screen and otherwise integrates the AS2 plug-in with ECXpert.

If your AS2 usage profile requires communication through a proxy server, the use of signature and encryption, or SSL, you need to make minor modifications to one or more fields in ECXpert's configuration file, ecx.ini. Appendix A, "ECXpert's Configuration File (ecx.ini)" describes the fields specific to AS2.

After completing the installation procedure, you can start the system components and configure partnerships for AS2 communications.

Starting and Stopping the Servers

Because the AS2 Communications Agents are a plug-in for 3.6.2, AS2 administration differs from administration for native Communications Agents (for example, FTP, SMTP, or ebXML). The AS2-related servers will not appear in the ECXpert Administrator interface, so these servers must be manually started and stopped.

Figure 1-1 on page 13 shows the servers involved for AS2 communication. They are the inbound and outbound AS2 Communication Agents, the HTTP Protocol listener that listens for inbound HTTP post messages, and the ECXpert server. The order in which you start the servers is important. If it is not done correctly, the

communication fails resulting in an error message. For example, if you brought up the web server before starting the AS2 server, when the servlet tries to deliver data to the AS2 Communications Agent, the server will not be available. The web server will return errors for inbound AS2 messages until the inbound server is started.

Starting the Servers

First, start the ECXpert system so that it is ready to send and receive AS2 data. Then start the AS2 inbound and outbound servers directly from the command line. Finally, start the Web Server instance to enable the HTTP/AS2 Protocol Listener.

➤ **To start the ECXpert server**

1. Start the main servers for ECXpert either through the command line or using the Administrative interface.

➤ **To start the AS2 Servers**

1. Start the AS2 inbound and outbound servers using the `as2-server start` command in the `$BDGHOME/http_as2/bin`.

   The listing in Table 1-4 shows this step for the example.

   The AS2 Server command-line interface allows you to manually start or stop the inbound and outbound AS2 servers. Note that there is no option to start or stop only one (inbound or outbound) of the servers.

Table 1-4 Example Listing for Starting the AS2 Servers

```
ecxadmin@host[74]% cd $BDGHOME/http_as2/bin
ecxadmin@host[75]% as2server start
ecxadmin@host[76]%

AS2 Server Bootstrap Loader v1.0
-> Created RMI registry.
-> Created Inbound AS2 Server.
-> Created Outbound AS2 Server.
-> Bootstrap Successful, Awaiting Requests...
```

➤ **To start the AS2 web server instance**

1. Start the web server instance for AS2 using the `start` command in `$IWS_SERVER_HOME/https-as2`.

➤ **To confirm that all servers are up and running**

- Use the console on Solaris systems to confirm that the Java™ AS2 servers are running correctly. For Windows systems, look at the output from the `as2server start` command and look at the `as2.log` file.
- Use the ECXpert Administrative interface to verify that ECXpert is running.
- Use the web server logs to verify that the web server is running.

Stopping the Servers

As with starting the servers, there is a specific order in which to stop each server in the system. After shutting the servers down, ECXpert cannot receive AS2 messages.

➤ **To stop the AS2 web server instance**

1. Stop the web server instance for AS2 using the `stop` command in `$IWS_SERVER_HOME/https-as2`.
2. Verify that the AS2 Inbound Communications Agent is not performing any work. One way to do this on Solaris machines is to use the `tail` command on the log files. On Windows machines, use the Task Manager.

   When there is no more activity, proceed to the next step.

➤ **To stop the AS2 Servers**

1. Stop the AS2 inbound and outbound servers using the `as2server stop` command in the `$BDGHOME/http_as2/bin`. See Table 1-5.

Table 1-5 Example Listing for Stopping the AS2 Servers

```
ecxadmin@host[68]% as2server stop
ecxadmin@host[69]%

AS2 Server Bootstrap Loader v1.0
-> Server shutdown in 5 seconds!
```

2. Verify that no work is currently in process on the ECXpert servers. When there is no activity, proceed to the next step.

➤ **To stop the ECXpert server**

1. Shut down the main ECXpert servers either through the graphical user interface or the command-line interface.

➤ **To confirm that all servers are shut down**

1. Confirm that all components have completed and are shut down by examining the AS2 logs and by confirming the processes are not in execution by running `ps -ef` commands in the Unix console.

Uninstalling AS2

The AS2 `setup.sh --uninstall` command restores the system to its preinstallation state.

➤ **To uninstall the AS2 plug-in components**

1. Execute the following command to uninstall the Sun ONE AS2 plug-in:

```
$BDGHOME/http_as2/setup.sh --unistall
```

NOTE Before uninstalling the AS2 plug-in, delete or change the outbound protocol for any partnerships that use the HTTP AS2 protocol. Failing to do this will leave the outbound partnership parameters in an inconsistent state.

What to Do Next

After you complete the AS2 plug-in installation and start the servers, you can test the new communications agents and set up encryption and signing if you need those features.

Test the Installation

Chapter 2, “Example: End-to-End Test for AS2 Messaging” on page 27 gives a detailed example for testing between sending and receiving hosts for a simple plain message. If you are familiar with ECXpert administration, you can skip that chapter. However, it is recommended that you become familiar with the fields of the new protocol screen that are now available in the ECXpert Administration interface. See Table 2-6 on page 36 and Figure 2-6 on page 37.

Set Up Encryption and Signing Features

To set up encryption and signing you must create certificates as described in Chapter 3, “Creating Certificates for AS2 Messaging” on page 45.

For instructions on setting up SSL, refer to Sun ONE Web Server documentation at http://docs.sun.com/db/coll/S1_ipwebsrvree60_en.

Chapter 2

Example: End-to-End Test for AS2 Messaging

This chapter provides a detailed example that shows how to set up ECXpert inbound and outbound hosts for AS2 messaging and steps through a procedure to send and receive an AS2 message. If you need more information to configure the trading relationship using ECXpert, refer to the ECXpert documentation set at docs.sun.com.

If you plan to follow the procedure in this chapter to test your AS2 setup, AS2 must be installed on two ECXpert machines—one to act as the outbound host or sender, the other to act as the inbound host or receiver. Chapter 1, "Getting Started with AS2 Messaging: Installation and Configuration" on page 11 gives the instructions for setting up the AS2 plug-in.

Example Overview

This example uses two members called partyA and partyB, which reside on the outbound host and the inbound host respectively. The purpose of the test is to ensure that you can send a plain messages end to end.

Setting up the Outbound Host

This section shows how to set up a partnership so that the outbound host can send AS2 messages. It contains procedures for the following tasks:

- Creating memberships for the sender and receiver
- Creating a partnership between the sender and receiver
- Creating a service list to route messages from partyA to partyB using the AS2 protocol

Perform these procedures using the ECXpert installation on the machine you are using for the sender.

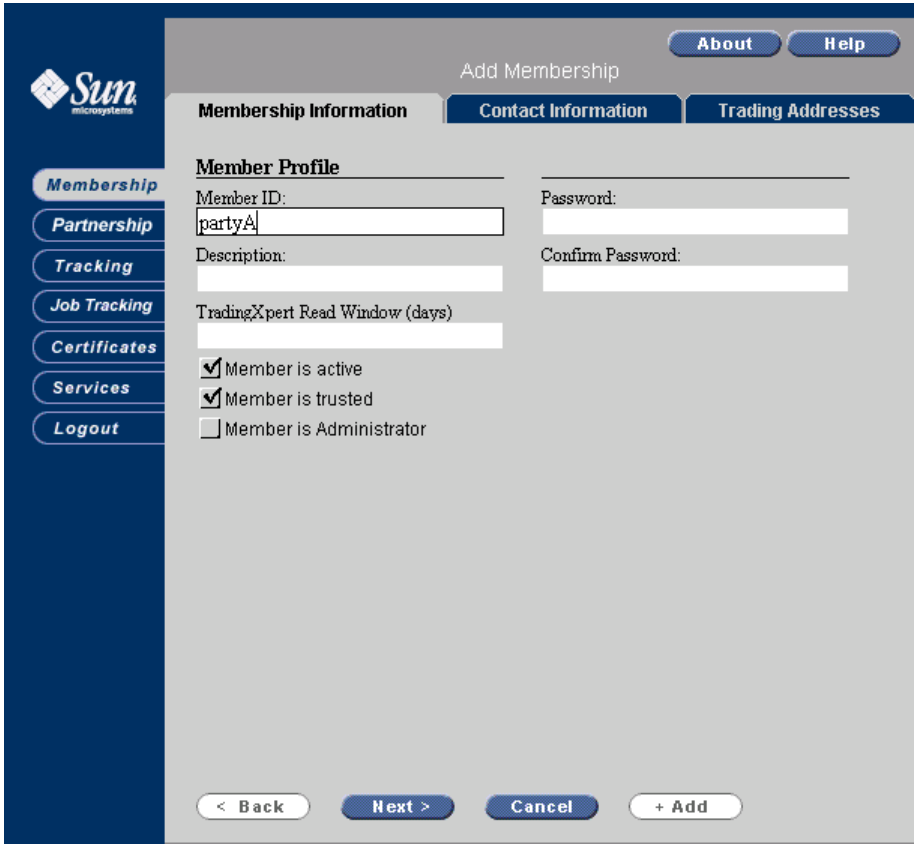
➤ **To create memberships**

1. Open the Support interface to ECXpert, and select Membership.
2. Select Add and specify the fields in Table 2-1, as shown in Figure 2-1:

Table 2-1 Field Values for Creating a Membership

Field	Value
Member ID	partyA
Select	Member is active
	Member is trusted
Password	If you do not specify a password, you must select the Member is trusted box.

Figure 2-1 Add a Member Screen

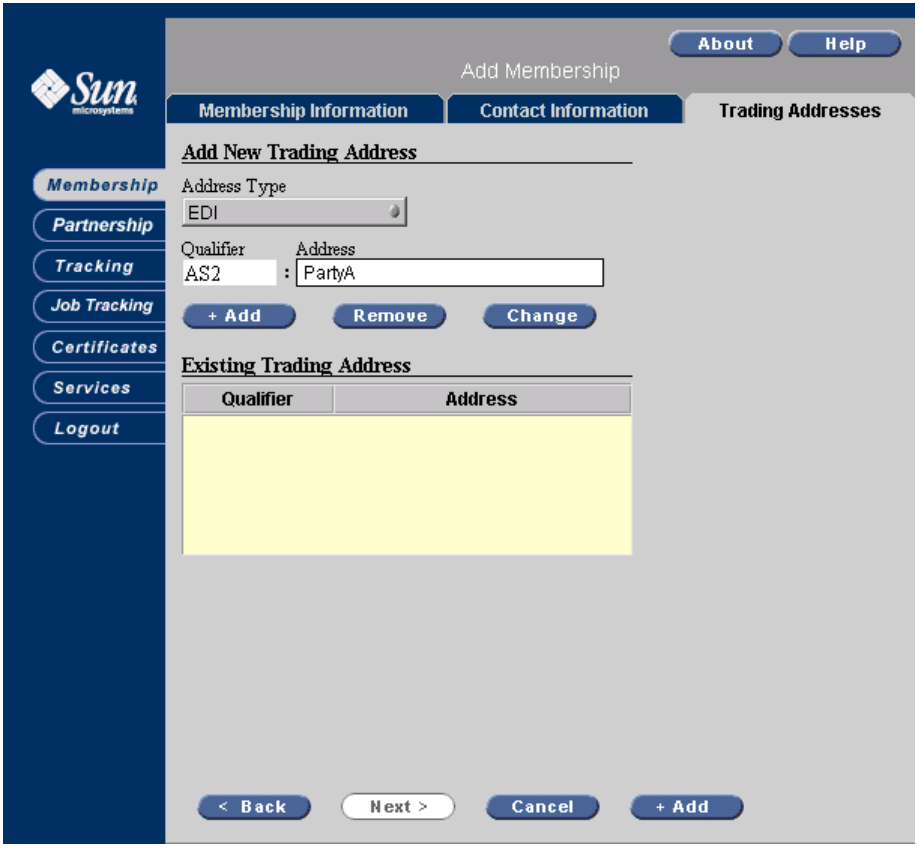


3. Select Next twice, bypassing the Contact Information page. These fields are not important for this test.
4. Specify information for the Trading Address as shown in the following table and Figure 2-2.

Table 2-2 Field Values for PartyA Address

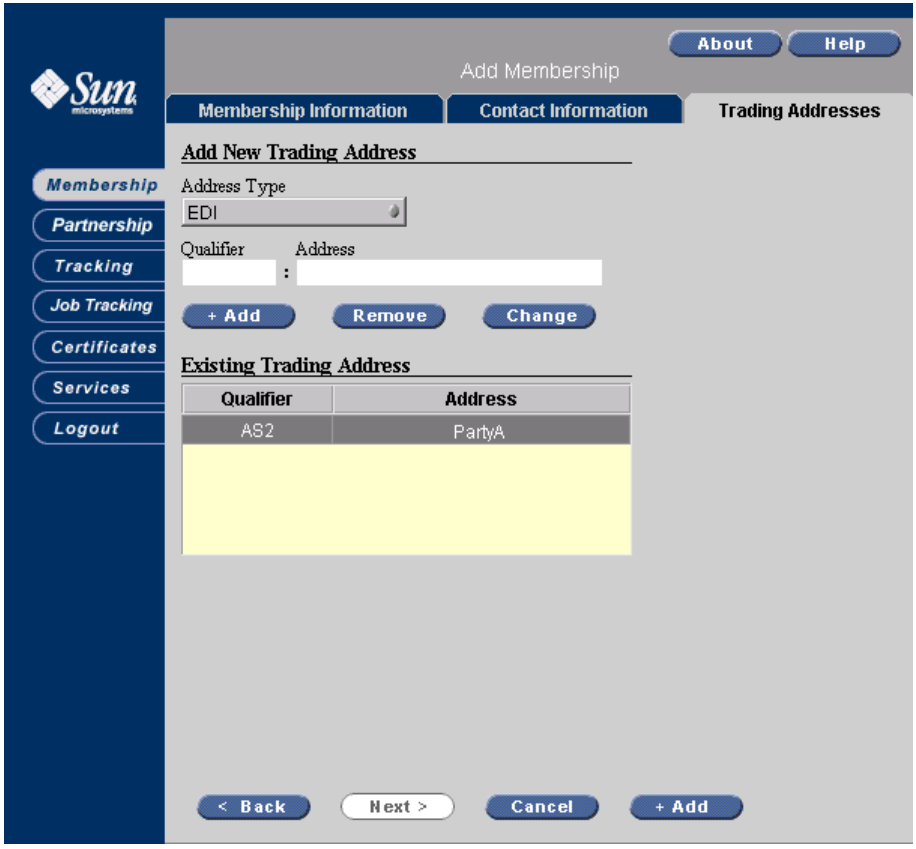
Field	Value
Address Type	EDI
Qualifier	AS2
Address	PartyA

Figure 2-2 Trading Address Screen



5. Click Change to modify the member with the new trading address. This results in the address being added to the Existing Trading Address list, as shown in Figure 2-3.

Figure 2-3 Trading Address Added Screen



6. Repeat Step 1 through Step 5 to add the Receiver member, with the following differences:

   a. In Membership Information, specify partyB for the Member ID.

   b. In Trading Addresses, specify the following:

Table 2-3 Field Values for ParyB Address

Field	Value
Address Type	EDI
Qualifier	AS2
Address	PartyB

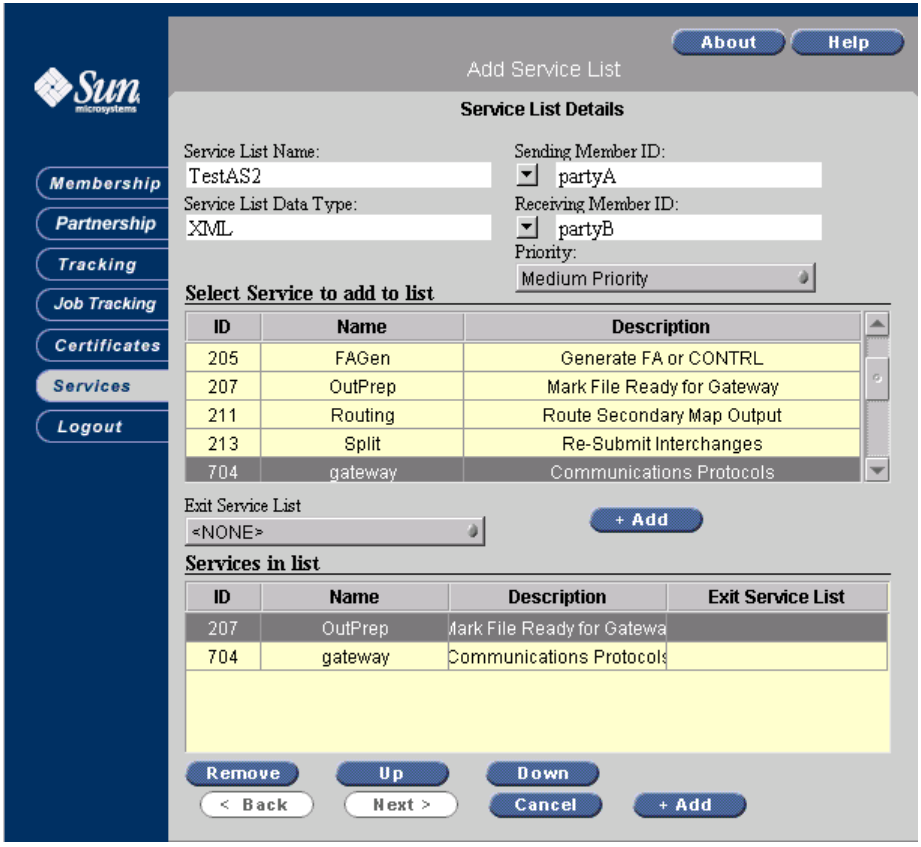
➤ **To create a service list**

1. From the Support interface, select Services.
2. Select Add and specify the following field values, as shown in Figure 2-4:

Table 2-4 Field Values for Creating a Service List

Field	Value
Service List Details	
Service List Name	TestAS2
Service List Datatype	XML
Sending Member ID	partyA
Receiving Member ID	partyB
Priority	Medium Priority
Exit Service List	<NONE>
Services in List	OutPrep
	gateway

Figure 2-4 Service List Screen



3. Click Add to add the Service List.

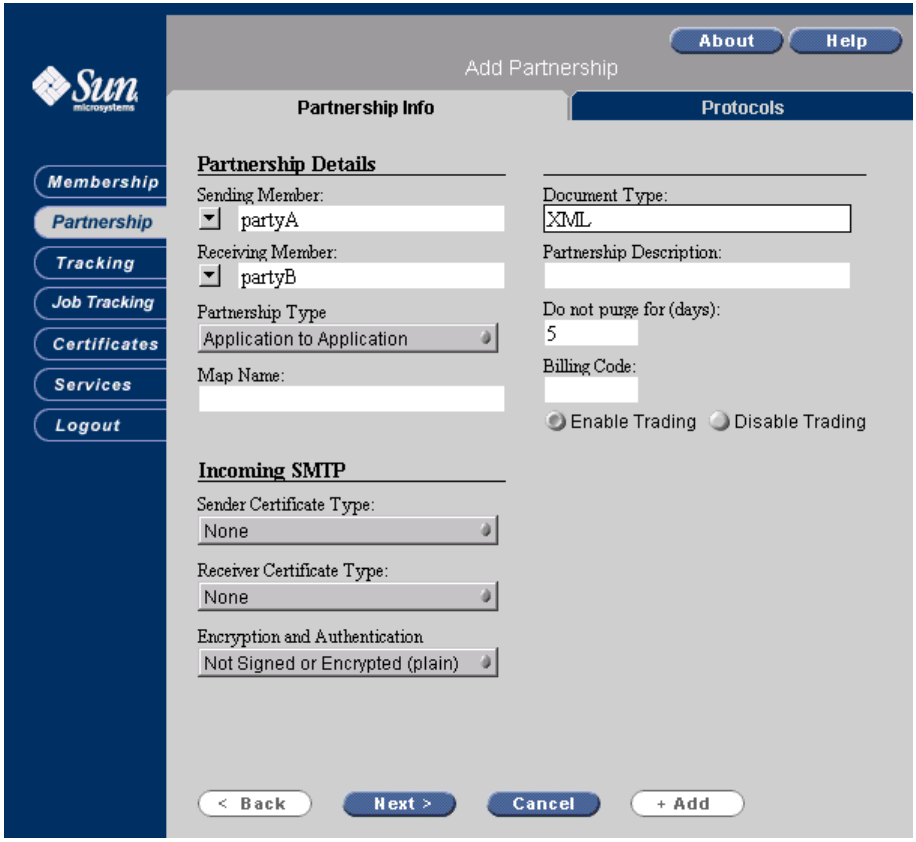
➤ **To create a partnership**

1. From the Support interface, select Partnership.
2. Select Add and specify the following field values:

Table 2-5 Field Values for Creating a Partnership

Field	Value
Partnership Details	
Sending Member	partyA
Receiving Member	partyB
Partnership Type	Application to Application
Map Name	<Leave blank>
Document Type	XML
Partnership Description	<Leave blank>
Do not purge	5
Billing Code	<Leave blank>
Enable Trading	On
Incoming SMTP	None None Not Signed or Encrypted (plain)

Figure 2-5 Add Partnership Screen



3. Select Next and specify field values for Protocols. In Table 2-6, the first line in each Value cell gives a specific field value for the example. A general definition of the field is also given.

Table 2-6 Fields Values for Outgoing Protocols

Field	Value
Outgoing Protocol	HTTP for AS2 The name for the Communications Agent is set in the `ecx.ini` file in the appropriate user-defined-x section at the User Visible Name field.
Pre-Communications Service	as2-pre-comm **NOTE**: You must specify as2-pre-comm or the communication session will fail.
Delivery Timing	Immediate **NOTE**: You must specify Immediate for this field or the communication session will fail.
Endpoint URL	`http://<`*inbound host*`>:<`*inbound port*`>/AS2/inbound` This is the destination to which the AS2 Outbound Server sends the HTTP POST.
Security	P You have the following options for this field: • P for plain messaging • S for signed messaging • E for encrypted messaging • B for signed and encrypted messaging
MDN	N You have the following options for this field: • N specifies no MDN acknowledgements • MP specifies plain (unsigned) MDNs • MS specifies signed MDNs
Synchronous Response	Y • Type Y to use the synchronous MDN return model. • Type N to use the asynchronous MDN return model.

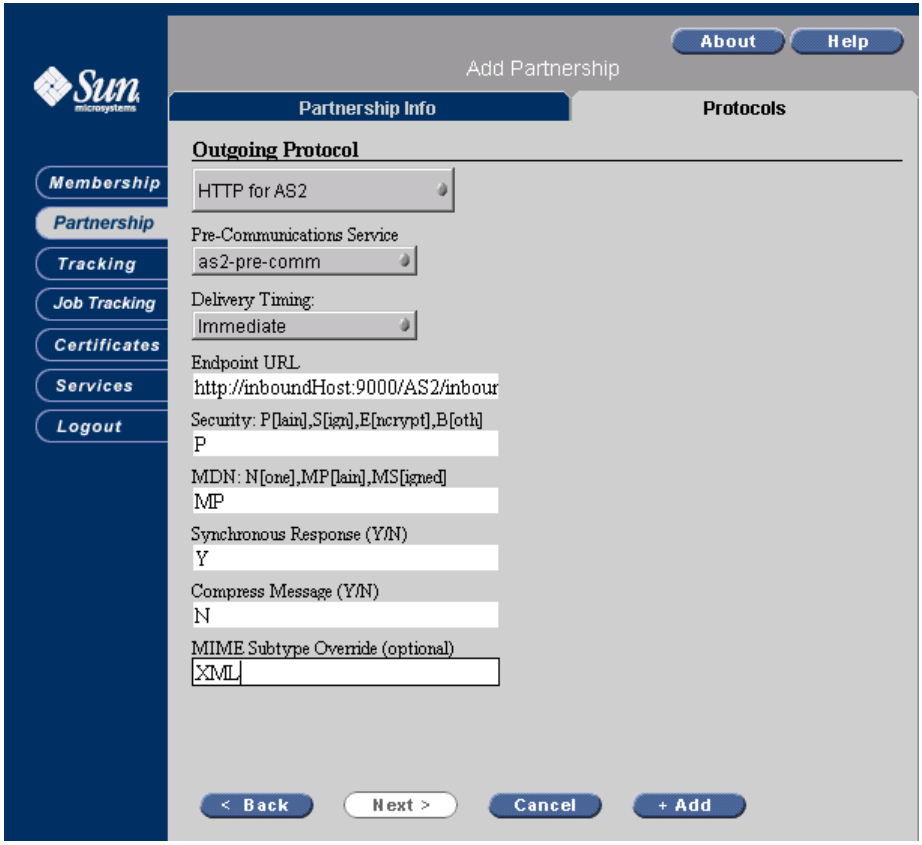
Table 2-6 Fields Values for Outgoing Protocols *(Continued)*

Field	Value
Compress Message	N • Type Y to request compression. • Type N for no compression.
MIME Subtype Override (Optional)	XML

Figure 2-6 shows the outgoing protocol specifications for this example.

If you need to use security or reliability, you must create certificates as described in Chapter 3, "Creating Certificates for AS2 Messaging" on page 45.

Figure 2-6 Outgoing Protocol Screen



Setting up the Inbound Host

This section shows how to set up a partnership so that the inbound host can receive AS2 messages. Much of it is identical to the last section for setting up the Outbound host. Differences are noted.

Perform these procedures using the ECXpert installation on the machine you are using for the receiver.

➤ **To create memberships**

1. Follow the same procedure and use exactly the same field information as shown in "To create memberships" on page 28.

➤ **To create a service list**

1. From the Support interface, select Services.
2. Select Add and specify the field values given in Table 2-7.

Table 2-7 Field Values for Creating a Service List

Field	Value
Service List Details	
Service List Name	AS2Test
Service List Datatype	XML
Sending Member ID	partyA
Receiving Member ID	partyB
Priority	Medium Priority
Exit Service List	<NONE>
Services in List	OutPrep
	gateway

3. Click Add to add the Service List.

To create a partnership follow the same procedure and use exactly the same field information as shown in "To create a partnership" on page 34 for Step 1 and Step 2.

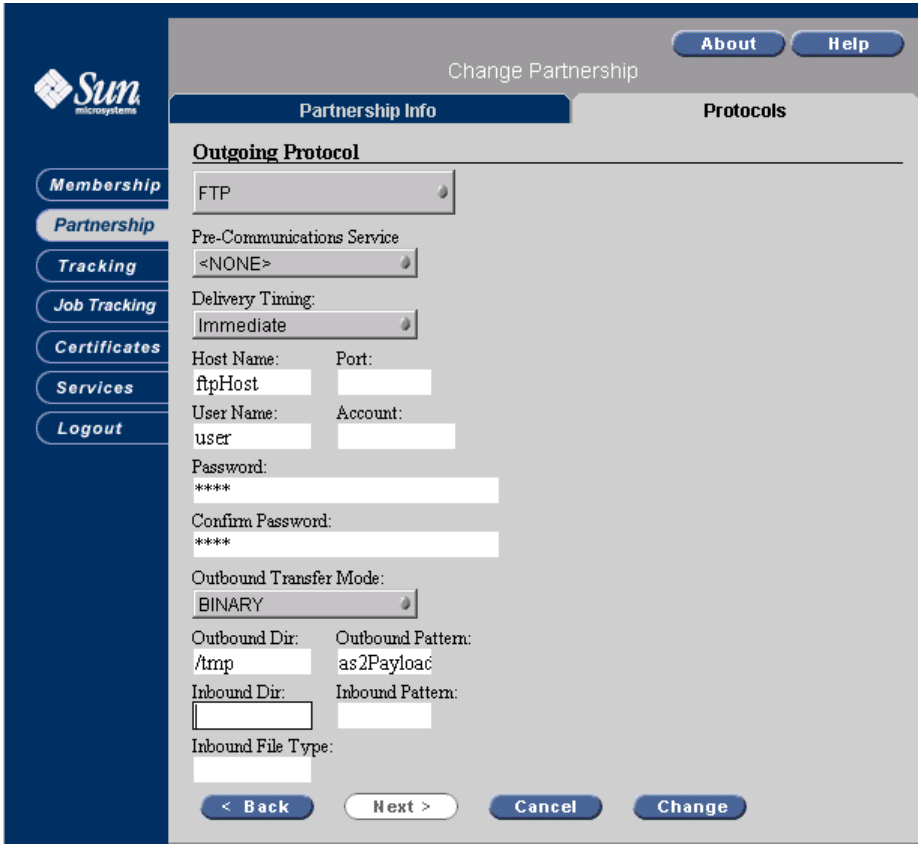
➤ **To create a partnership**

1. Select Next and specify the following for Protocols for the example, as shown in Figure 2-7:

Table 2-8 Field Values for Creating a Partnership

Field	Value
Outgoing Protocol	FTP
Pre-Communications Service	<NONE>
Delivery Timing	Immediate
Host Name	ftpHost
Port	<Leave blank>
User name	The name of the ECXpert user
Account	<Leave blank>
Password	The password for the ECXpert user
Outbound Transfer Mode	BINARY
Outbound Dir	`/tmp`
	Create a temporary output directory and provide path here.
Output Pattern	outBoundMsg.xml
	This is the name of the output file.
Inbound Dir	<Leave blank>
Inbound Pattern	<Leave blank>
Inbound File Type	<Leave blank>

Figure 2-7 Receiver's Protocol Screen



2. Click Change.

Modifying the ecx.ini Configuration File

You must modify the ecx.ini file to identify the sender and receiver to ECXpert.

➤ **To modify the ecx.ini file**

1. Open the `ecx.ini` using a text editor.
2. Locate the AS2 Server section and modify the following fields:

   ```
   submit_sender = partyA
   submit_receiver = partyB
   submit_file_type = XML
   ```

   Leave `submit_password` blank.
3. Save and exit the `ecx.ini` file.
4. Shut down and restart the AS2 Server. See "Starting and Stopping the Servers" on page 21 for instructions.

Sending a Document and Checking Results

Use the submission utility to submit a document from the outbound host, partyA.

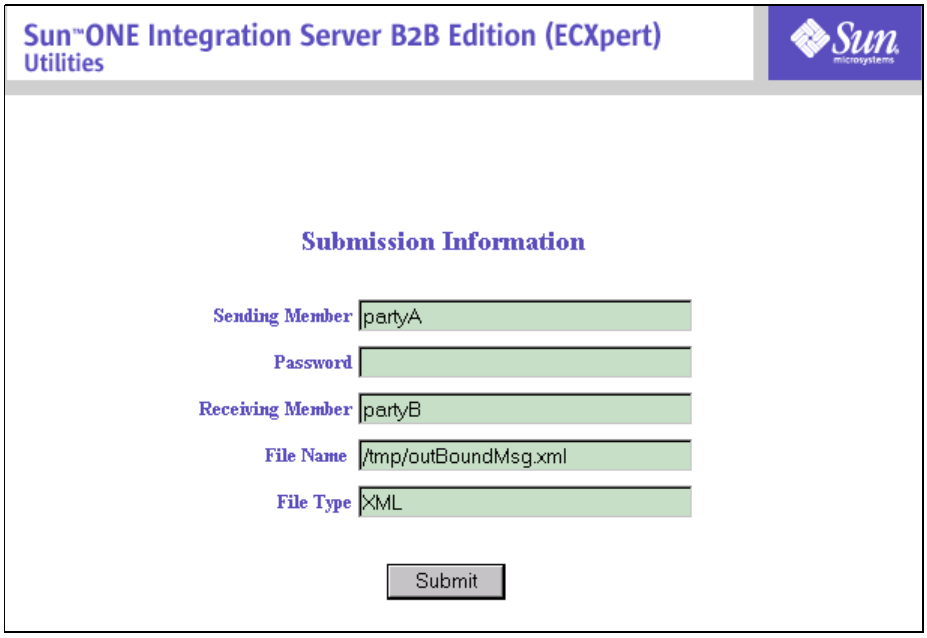
➤ **To submit a document**

1. Open the ECXpert submission utility and specify the field values in Table 2-9:

Table 2-9 Field Values for Submitting a Document

Field	Value
Sending Member	partyA
Receiving Member	partyB
File Name	`/tmp/outBoundMsg.xml` (or whatever you called the output file and your temporary directory)
File Type	XML

Figure 2-8 Submitting a Document Screen



➤ **To check the results**

1. Record the tracking ID and view the results in ECXpert's event log from both the sending and the receiving host machines.

   The following figures show successful submission for the sender and receiver.

Figure 2-9 Event Log Screen

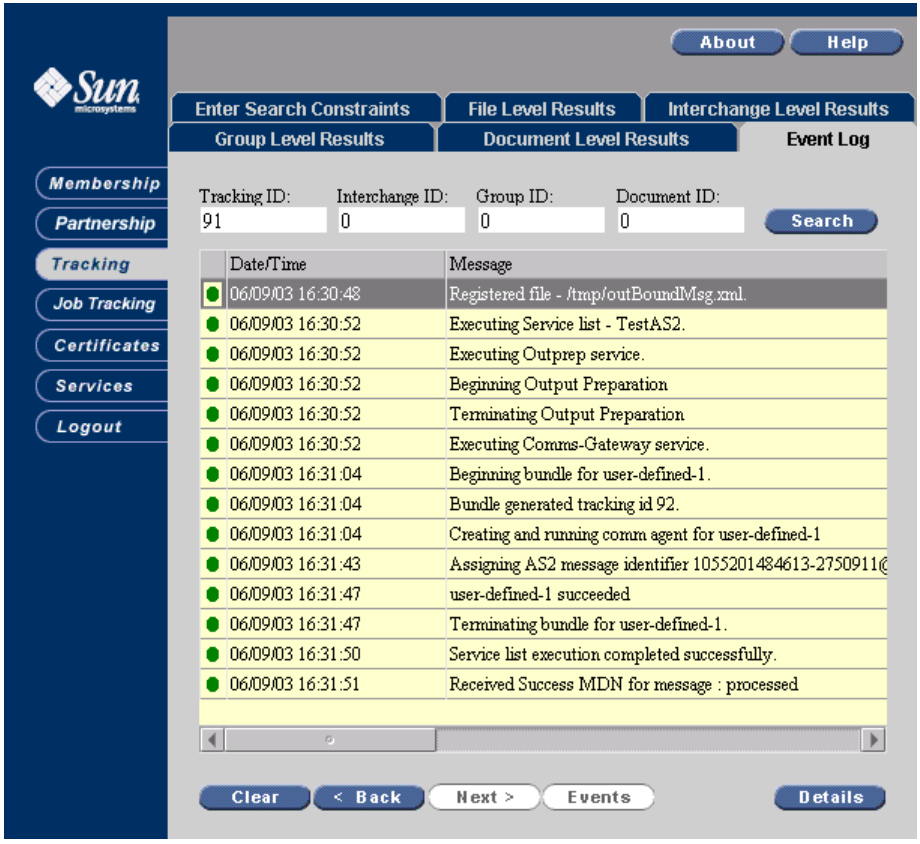
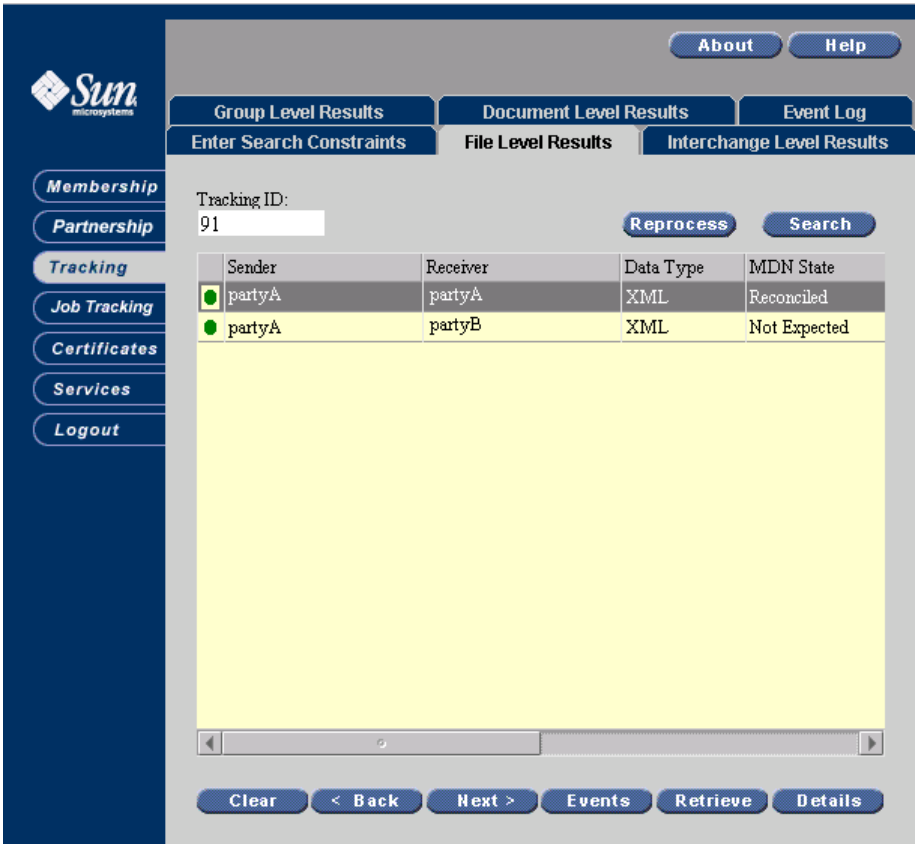


Figure 2-10 File Level Results Screen



Chapter 3

Creating Certificates for AS2 Messaging

This chapter explains how to use the `keytool` utility to acquire certificates from a third party certificate authority (CA) such as Verisign or Thawte. You also use `keytool` to import, export, and delete certificates. `keytool` is a security tool available with the Java™ 2 Software Development Kit (Java 2 SDK), Standard Edition that is used with this release.

This chapter covers the following topics:

- "Introduction to the keytool Utility" describes relevant options and output for the `keytool` utility.
- "Example: Managing Keystores for AS2" provides a detailed example of the steps necessary to enable secure messaging between two AS2 trading partners.

Introduction to the keytool Utility

This section describes relevant options and output that you need to be familiar with when using `keytool` to manage a keystore for secure AS2 communication. For more detailed information about the `keytool` utility, see the `docs` directory of your Java 2 SDK installation. You may have to download the Java 2 SDK documentation separately. This documentation is also available online at the following URL:

`http://java.sun.com/j2se/1.4/docs/tooldocs/solaris/keytool.html`

Creating a Keystore with keytool

Use the `-genkey` option with `keytool` to generate a keystore (private key). Typically, a `keystore` file has a `.jks` extension.

When generating the keystore you specify an alias and a name.

The *alias* you specify must match the AS2 party identifier (i.e. address) for the member that will use the key or certificate.

The *name* is an X.500 distinguished name and has the following format:

```
CN=cName, OU=orgUnit, O=org, L=city, S=state, C=countryCode
```

The CN you specify should be a fully qualified domain name of the host you are using for an ECXpert secure endpoint (for example, `host.sun.com`). The CN can also be an IP address. Check with the CA you are using to determine their requirements for CN names.

CAUTION Make sure your `keystore` file, which contains your private key, is kept within a secure location on your system.

Issuing Certificate Requests

To issue certificate requests, which you send to the appropriate Certification Authority, use the `-certreq` option with `keytool` to generate a Certificate Signing Request (CSR) file that you send to a CA. The CA authenticates the certificate requestor and replies with a public certificate and one or more certificates that are trusted.

Typically, a certificate request file has a `.csr` extension. The CA returns certificate files with a `.cer` extension. One of the returned certificates is the public key that matches the private key created with the `keytool -genkey` command. The other certificate files are trusted certificates that verify your public key.

Importing Trusted Certificates

Import trusted certificates if the certificates obtained from the CA have not previously been imported into the truststore supplied with your Java 2 SDK (`cacerts` file).

Use the `-import` option to import the trusted certificates returned by the CA into the `cacerts` file.

Designating Public Keys for Other Parties as Trusted

Use the `-trustcacerts` option with the `-import` option to provide a chain of trust to your `keystore` file. You must specify your keystore alias and keystore password when doing this step.

Verifying that the Key Pair Match

Use the `-printcert` option to `keytool` to print information in your public certificate. Then use the `-list` option to print information in your private key. You can then compare the certificate fingerprints in the listings, as shown in the example of Table 3-1.

Table 3-1 Example Listing for Public and Private Key File Verification

```
% keytool -printcert -file as2.cer
Owner: CN=host.sun.com, OU=SQA, O=Sun, L=Pheonix, ST=AZ, C=US
Issuer: OU=For VeriSign authorized testing only. No assurances
(C)VS1997,
OU=www.verisign.com/repository/TestCPS Incorp.
By Ref. Liab. LTD., O="VeriSign, Inc"
Serial number: 29415e9496b158ad7d040f6d45b0aac4
Valid from: Thu Dec 05 16:00:00 PST 2002 until: Fri Dec 20
15:59:59 PST 2002
Certificate fingerprints:
         MD5:  CC:2A:56:62:FF:8A:1A:87:7E:D0:D1:D8:28:54:E4:F2
         SHA1:
27:C6:BF:AE:18:B3:D9:2A:91:2B:7E:F2:3F:64:88:DF:00:A1:3E:28

% keytool -list -keystore as2.jks -storepass PartyA

Keystore type: jks
Keystore provider: SUN

Your keystore contains 1 entry

sfbay.sun.com, Dec 6, 2002, keyEntry,
Certificate fingerprint (MD5):
CC:2A:56:62:FF:8A:1A:87:7E:D0:D1:D8:28:54:E4:F2
```

Example: Managing Keystores for AS2

This section provides a detailed example of the steps necessary to manage secure messaging between two AS2 agents. This involves creating key pairs and importing and exporting certificates. You might also need to delete certificates, which you can also do using the `keytool` utility.

The steps illustrated in this section are as follows:

1. Create new private/public key pair for partyA, Step 1 on page 49.
2. Export a self-signed X.509 certificate from partyA's keystore, Step 2 on page 50.
3. Create a private/public key pair for partyB, Step 3 on page 51.
4. Export a self-signed X.509 certificate from partyB's keystore, Step 4 on page 52.
5. Import partyB's certificate into partyA's keystore, Step 1 on page 53.
6. Import partyA's certificate into partyB's keystore, Step 2 on page 53.
7. Modify the AS2 configurations to use the new keystores, "Modifying the AS2 Configurations for Keystore Use" on page 54.
8. Stop and restart the AS2 Servers, Step 3 on page 55.

The example in this section uses the two trading partners created in Chapter 2, "Example: End-to-End Test for AS2 Messaging".

Preconditions

Ensure that the following preconditions are met before proceeding:

- The JDK/JRE version is 1.4 or later. You can execute the `java -version` command to check the version.
- The PATH environment variable points to your JDK/JRE 1.4 installation. If it does not, edit the appropriate configuration file or use the `setenv PATH` command on Solaris machines. On Windows, set the environment variable using the Advanced tab of the Control Panel's System Tool.
- You have created a directory in which to store the new certificates and keys. Execute the procedures in this section so that the certificates and keys are created within this directory.

Creating Key Pairs and Exporting Certificates

The following steps create and export key pairs for each of two trading partners taking part in secure AS2 communication.

1. Create a new private/public key pair for partyA.

   This step assumes you do not have an existing keystore and creates a new one. Typically, you create additional key pairs within an existing keystore. The `keytool` utility automatically creates a keystore if one does not exist. Execute `keytool` from the partyA host machine.

   After you issue the `keytool` command, the utility prompts for values. The combination of all the values should be unique across all keys and certificates in the keystore. The easiest way to ensure that they are unique is to specify a unique value set for first and last name parameter, which is used for the CN attribute in the X.509 certificate. Also make sure that the value supplied for the alias parameter is the same as the as2 identifier used for the ECXpert member.

   The example session listed in Table 3-2 illustrates the use of the `keytool` command to create a 1024 bit key for RSA encryption. In this example, the AS2 identifier (PartyA) is used as the alias—this is the value specified as the ECXpert trading address in Figure 2-2 on page 30, Trading Address Screen.

Table 3-2 Example Listing for Key Generation for partyA

```
ecxadmin@host% keytool -genkey -v -alias "PartyA" -keyalg RSA
-keysize 1024 -keystore PartyA.jks
Enter keystore password: password
What is your first and last name?
[Unknown]: PartyA Message Security
What is the name of your organizational unit?
[Unknown]: PartyA
What is the name of your organization?
[Unknown]: PartyA
What is the name of your City or Locality?
[Unknown]: Phoenix
What is the name of your State or Province?
[Unknown]: AZ
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=PartyA Message Security, OU=PartyA, O=PartyA, L=Phoenix,
ST=AZ, C=US correct?
[no]: y
Generating 1,024 bit RSA key pair and self-signed certificate
(MD5WithRSA) for: CN=PartyA Message Security, OU=PartyA,
O=PartyA, L=Phoenix, ST=AZ, C=US
Enter key password for <PartyA>
(RETURN if same as keystore password):
[Saving PartyA.jks]
```

2. Export a self-signed X.509 certificate from partyA's keystore.

   Use the `keytool` command to export a self-signed certificate based on the public/private key pair generated in Step 1 on page 48. The parameters used in the `keytool` listing example of Table 3-4 are defined as follows in Table 3-3.

Table 3-3 Flag descriptions for the `keytool` command

Flag	Description
`alias`	The alias name for the certificate you want to export.
`rfc`	Output the certificate with base64 encoding.
`file`	The name of the file you want to output to (it is useful to use a .cer suffix for viewing the certificate in a Windows environment).
`keystore`	The keystore you are exporting from.

Table 3-4 Example Listing for Exporting a Certificate from partyA's Keystore

```
ecxadmin@host% keytool -export -v -rfc -alias "PartyA" -file
PartyA.cer -keystore PartyA.jks
Enter keystore password: password
Certificate stored in file <PartyA.cer>
```

After `keytool` exits, you can send the exported file (PartyA.cer in this example) to the trading partner (partyB). partyB uses the certificate to verify partyA's signature and to encrypt messages that it sends back to partyA.

3. Create a private/public key pair for partyB.

   Repeat Step 1 on page 48 on the partyB host machine, substituting the appropriate parameters (PartyB for PartyA, and so on). Be sure to use the correct AS2 party ID for the key alias.

   If you are creating the keystores on the same machine (and not each on a different machine), use two distinct names for the keystores. Although this example does use two distinct machines, it explicitly illustrates the use of two names for the keystores, `PartyA.jks` and `PartyB.jks`.

   The example session in Table 3-5 illustrates key generation for partyB. Make sure that you use the trading partner's AS2 identifier as the alias name and respond with `y` when `keytool` prompts `Trust the certificate?`.

Table 3-5 Example Listing for Key Generation for partyB

```
ecxadmin@kite% keytool -genkey -v -alias "PartyB" -keyalg RSA
-keysize
1024 -keystore PartyB.jks
Enter keystore password: password
What is your first and last name?
[Unknown]: PartyB Message Security
What is the name of your organizational unit?
[Unknown]: PartyB
What is the name of your organization?
[Unknown]: PartyB
What is the name of your City or Locality?
[Unknown]: Phoenix
What is the name of your State or Province?
[Unknown]: AZ
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=PartyB Message Security, OU=PartyA, O=PartyA L=Phoenix,
ST=AZ, C=US correct?
[no]: y
Generating 1,024 bit RSA key pair and self-signed certificate
(MD5WithRSA) for: CN=PartyB Message Security, OU=PartyB,
O=PartyB, L=Phoenix,
ST=AZ, C=US
Enter key password for <PartyB>
(RETURN if same as keystore password):
[Saving PartyB.jks]
```

4. Export a self-signed X.509 certificate from partyB's keystore.

   This step is similar to Step 2 on page 50. The difference is that you substitute values for PartyB.

Table 3-6 Example Listing for Exporting a Certificate from partyB's Keystore

```
ecxadmin@host% keytool -export -v -rfc -alias "PartyB" -file
PartyB.cer -keystore PartyB.jks
Enter keystore password: password
Certificate stored in file <PartyB.cer>
```

Importing Certificates

1. Import partyB's certificate into partyA's keystore.

   This step imports partyB's certificate (PartyB.cer) into the partyA keystore. Execute the `keytool -import` command from the partyA host machine. You must have access to the trading partner's X.509 certificate and be sure to use the trading partner's AS2 identifier as the alias name. Table 3-7 shows how to use keytool to import the certificate.

   Table 3-7 Listing for Importing partyB's Certificate into partyA's Keystore

```
ecxadmin@host% keytool -import -v -alias "PartyB" -file
PartyB.cer -keystore PartyB.jks
Enter keystore password: password
Owner: CN=PartyB Message Security, OU=PartyB, O=PartyB,
L=Phoenix, ST=AZ, C=uS
Issuer: CN=PartyB Message Security, OU=PartyB, O=PartyB,
L=Phoenix, ST=AZ, C=US
Serial number: 3ee6b6f5 Valid from: Tue Jun 10 21:58:29 PDT
2003 until: Mon Sep 08 21:58:29 PDT 2003
Certificate fingerprints:
MD5: AD:40:07:9A:8B:A4:7F:D7:95:BB:73:00:5A:A4:2B:9A
SHA1: 66:3E:88:14:DE:1E:A9:9B:AF:80:29:93:45:
A2:71:5B:1C:29:15:D8
Trust this certificate? [no]: y
Certificate was added to keystore
[Saving PartyB.jks]
```

2. Import partyA's certificate into partyB's keystore.

   This step is similar to Step 1 on page 53. The difference is that you substitute values for PartyB and execute the commands from the partyB host.

   Make sure that you use the trading partner's AS2 identifier as the alias name and respond with `y` when keytool prompts `Trust the certificate?`.

Table 3-8 Listing for Importing partyA's Certificate into partyB's Keystore

```
ecxadmin@kite% keytool -import -v -alias "PartyA" -file
PartyA.cer -keystore as2.jks
Enter keystore password: password
Owner: CN=PartyA Message Security, OU=PartyA, O=PartyA,
L=Phoenix, ST=AZ, C=US
Issuer: CN=PartyB Message Security, OU=PartyA, O=PartyA,
L=Phoenix, ST=AZ, C=US
Serial number: 3ee6b6f5
Valid from: Tue Jun 10 21:58:29 PDT 2003 until: Mon Sep 08
21:58:29 PDT 2003
Certificate fingerprints:
MD5: AD:40:07:9A:8B:A4:7F:D7:95:BB:73:00:5A:A4:2B:9A
SHA1: 66:3E:88:14:DE:1E:A9:9B:AF:80:29:93:45:
A2:71:5B:1C:29:15:D8
Trust this certificate? [no]: y
Certificate was added to keystore
[Saving as2.jks]
```

Modifying the AS2 Configurations for Keystore Use

You must modify the `ecx.ini` file for each AS2 installation so that the AS2 server recognizes the new keystore. This requires adding the following information:

- The path to the keystore
- The password used to access that keystore

For security purposes, the keystore password must be encrypted within the `ecx.ini` file. The following steps describe the process.

1. Run the `as2passwd` utility in `$BDGHOME/http_as2/bin` to encrypt the keystore password as shown in the following example. Copy the resulting encrypted text into the `ecx.ini` file in the next step.

Table 3-9 Example Listing for `as2passwd`

```
ecxadmin@host% as2passwd
Text to encrypt -> password
Encrypted Text -> MbNRPeMoThN4TUTCoFoEmZoCNXUFm5pq
```

2. Use a text editor to update the `keystore_file` and `keystore_pass` fields under the server section of the `ecx.ini` file on each machine as follows:

Table 3-10 Field Values for ecx.ini Security Fields

ecx.ini Field	Value
`keystore_file`	The fully qualified path to the keystore.
`keystore_pass`	The encrypted password for the keystore.

See Appendix A, "ECXpert's Configuration File (ecx.ini)" for information about customizing the `ecx.ini` file for AS2.

The values for these fields for the example are as follows:

```
keystore_file = /export/home/actraadm/as2/PartyA.jks
```

```
keystore_pass = MbNRPeMoThN4TUTCoFoEmZoCNXUFm5pq
```

3. Stop and start the AS2 server instance on each host for which you modified the `ecx.ini` file. See "Starting and Stopping the Servers" on page 21 for detailed instructions on server startup and shutdown.

Appendix A

ECXpert's Configuration File (ecx.ini)

AS2 processing uses configuration information from the ECXpert configuration file, `ecx.ini`. The AS2 plug-in installer prompts for all required information and automatically configures the parameters within the file. However, some AS2 usage profiles require that the user modify `ecx.ini` as a post-installation task. These are:

- Communication through a proxy server
- Signatures, encryption, or both
- SSL

The AS2 section in `ecx.ini` is located under the `[as2-server]` label.

This appendix provides a description of each field in the AS2 section. Mandatory fields are identified with italics.

Table A-1 Field descriptions for the AS2 section of `ecx.ini`

Field	Description
visible_name	The name used to display this server section in the ECXpert user interface.
server_port	The port under which the RMI registry will run. The default is 9099. It can be changed if the port is already bound to another process or if you want to select a different port.
in_server_name	The name under which the inbound server's remote object reference will be bound in the RMI registry. It is set by the installer and should not be changed by an administrator.
out_server_name	The name under which the outbound server's remote object reference is bound in the RMI registry. It is set by the installer and should not be changed by an administrator.

Table A-1 Field descriptions for the AS2 section of `ecx.ini` *(Continued)*

Field	Description
`submit_sender`	Hard-codes the sender name for submissions to ECXpert. Use this to submit all AS2 documents as the same user. If no value is specified, the AS2 Server determines the sending member's name based on its AS2 party ID and submits the document using that value.
`submit_receiver`	Hard-codes the receiver name for sending from ECXpert. Use this to send all AS2 documents to the same user. If no value is specified, the AS2 Server determines the receiving member's name based on its AS2 party ID and sends the document using that value.
`submit_password`	Password used for submission. If not specified, the sending member must be a trusted member in ECXpert, because no password will be passed in the submission. If you specify a password, it must be encrypted using the as2passwd utility.
`submit_file_type`	File type used for submission. If not specified, the MIME sub-type specified in the `Content-Type` header of the AS2 message is used. If the sub-type is EDI-X12, EDIFACT, or EDI-Consent, the value is trimmed to 'EDI'. Examples are: Content-Type Submission File Type application/XML XML application/EDI-X12 EDI application/EDIFACT EDI application/idox-x8-41 idox-x8-41
`http_proxy_host`	The hostname of the proxy server (if used).
`http_proxy_port`	The port of the proxy server (if used).
log_level	The level of granularity used for log file entries. Possible values include: SEVERE, WARNING, INFO, FINE, FINER, FINEST.
`keystore_file`	If the system is using any level of message security (encryption, digital signature) and/or SSL, the administrator must specify a keystore file. The file path must point to an existing Java keystore.
`keystore_pass`	The password used to access the keystore. Must be encrypted using the as2passwd utility.
`admin_user`	The ECXpert user account used to query and update information in ECXpert.
`admin_pass`	The password for the ECXpert admin user. Must be encrypted using the as2passwd utility.
root_dir	The AS2 root directory.

Table A-1 Field descriptions for the AS2 section of `ecx.ini` *(Continued)*

Field	Description
`asynch_url`	A local URL used as a return address when requesting asynchronous MDNs.
jdbc_uri	The JDBC hostname/port/SID used by the JDBC driver.
ssl_host_verify	Set to true if the server name of an SSL host should be checked against the credentials presented in its certificate. False is the default.
`custom_mdn_path`	The path to a user-defined message for AS2 MDNs. When present, the value for this field should point to a file containing text that is used in the second part of an MDN multipart message (for example, `The user section`). If a path is not supplied, or if it is unresolvable or unreadable, the following default MDN text will be used: `AS2 Message Disposition Notification`

Index

D

E

F

H

I

J

K

L

M

O

P

R

S

T

U

V

W