Forte Fusion Backbone System Guide

Release 2.1 of Forte Fusion™

Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 U.S.A. 1-800-786-7638

Part No. 806-7136-01
November 2000, Revision A

Contents

Fusion Script Commands *(continued)*

Preface

The Forte Fusion Backbone is a suite of business integration tools based on XML (Extensible Markup Language), and XSLT (Extensible Stylesheet Language for Transformations). Fusion supports different styles of business integration across heterogeneous components by providing advanced message brokering and data transformation services, as well as enterprise-class process automation for integrated applications that participate in business processes.

This manual, the *Forte Fusion Backbone System Guide*, explains the architecture of Forte Fusion, important concepts and functions of the Fusion application proxy, and how to configure and administer a Fusion backbone. It includes an overview of the XML documents processed by the Fusion application proxy. The appendices provide a detailed reference of the Fusion Script commands used to manage the backbone.

This book assumes some familiarity with Forte system management concepts, the Fusion process engine, and an understanding of either HTTP protocol or the JMS interface, XML, and XSLT. The *Forte Fusion Glossary* published on the CD-ROM is intended as a useful reference.

About Forte Fusion

Forte Fusion is a suite of business integration tools for integrating and coordinating heterogeneous applications. The tools and software components provided with Forte Fusion let you integrate newly developed applications, legacy applications, and off-the-shelf packages into business processes that are automated and controlled by a process engine.

A Fusion system is a set of tools and software modules installed on top of a compatible version of the Forte Application Environment. It is composed of two subsystems, a process management system and an XML-based backbone system.

Fusion Process Management System The Fusion Process Management System (formerly known as Conductor) provides a set of tools and software modules that support the development, execution, and management of business processes. The heart of this system is the Fusion Process Engine, which controls and manages business processes from beginning to end, coordinating the work of the different resources or applications that participate in the processes.

Forte Fusion customers use the Fusion Process Management System to:

- develop process logic with the graphical process development workshops
- manage sessions and processes, and the engine itself, using the Fusion Console and other tools
- build applications, called process clients, that make direct API calls to the process engine, using the process client APIs (Forte 4GL, CORBA/IIOP, JavaBeans, ActiveX, or C++)

Backbone System The Fusion Backbone System provides a set of tools and software modules that use XML messaging over HTTP or JMS to simplify communication and coordination between applications. A Fusion backbone can support different styles of integration, but the backbone is always installed on top of the Fusion process engine runtime. The heart of a backbone system is a set of application proxies that perform message brokering and data transformation on behalf of applications. For business process support, proxies interact with the Fusion process engine on behalf of any applications that participate in a common business process. The main purpose of these interactions is to communicate the initiation and completion of work activities.

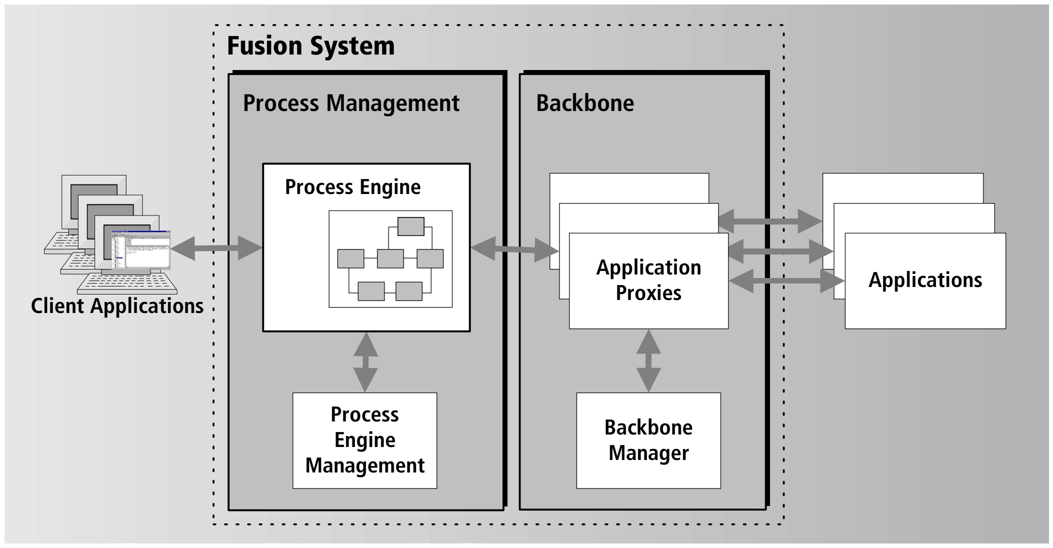

Figure 1 *Fusion System and Subsystems*

Forte provides adapters as well as an adapter toolkit to integrate packages or custom applications that lack a native XML/HTTP interface into a Fusion backbone.

Forte Fusion customers use the Fusion Backbone System primarily to:

- provide an XML/HTTP interface between proxies and applications
- configure application proxies to participate in a managed business process

The XML/XSL Workshops provided with Fusion facilitate the development, testing, debugging, storage, and management of sample XML documents and the XSL stylesheets used for message transformation between applications.

Organization of this Manual

This manual begins with an architectural overview of the Forte Fusion Backbone and then explains important concepts and features of the Fusion application proxy. The next chapter explains how to configure and manage backbones and proxies. A conceptual overview of the XML documents processed by the proxy is included. The appendices explain the Fusion Script utility and provide a full command reference. Briefly, the chapters are:

Chapter	Description
Chapter 1, “Introduction”	Explains the Forte Fusion Backbone architecture, the styles of integration it supports, and typical integration tasks.
Chapter 2, “Proxy Concepts and Features”	Explains concepts and features of the Fusion application proxy.
Chapter 3, “Backbone System Configuration and Administration”	Explains how to configure and manage a Fusion backbone system.
Chapter 4, “Proxy Documents”	Explains the structure and vocabulary of the XML documents that the proxy processes.
Appendix A, “Fusion Script Commands”	Provides an administrative command reference for the Fusion Backbone system management script utility.
Appendix B, “General Script Commands”	Provides a command reference for general commands that may be useful when administering a Fusion system.

Conventions

This manual uses standard Forte documentation conventions in specifying command syntax and in documenting Forte 4GL TOOL code.

Command Syntax Conventions

The specifications of command syntax in this manual use a “brackets and braces” format. The following table describes this format:

Format	Description
bold	Bold text is a reserved word; type the word exactly as shown.
italics	Italicized text is a generic term that represents a set of options or values. Substitute an appropriate clause or value where you see italic text.
UPPERCASE	Uppercase text represents a constant. Type uppercase text exactly as shown.
underline	Underlined text represents a default value.
vertical bars |	Vertical bars indicate a mutually exclusive choice between items. See braces and brackets, below.
braces { }	Braces indicate a required clause. When a list of items separated by vertical bars is enclosed in braces, you must enter one of the items from the list. Do not enter the braces or vertical bars.
brackets [ ]	Brackets indicate an optional clause. When a list of items separated by vertical bars is enclosed by brackets, you can either select one item from the list or ignore the entire clause. Do not enter the brackets or vertical bars.
ellipsis …	The item preceding an ellipsis may be repeated one or more times. When a clause in braces is followed by an ellipsis, you can use the clause one or more times. When a clause in brackets is followed by an ellipsis, you can use the clause zero or more times.

Forte 4GL TOOL Code Conventions

Where this manual includes documentation or examples of Forte 4GL TOOL code, the TOOL code conventions in the following table are used.

Format	Description
parentheses ( )	Parentheses are used in TOOL code to enclose a parameter list. Always include the parentheses with the parameter list.
comma ,	Commas are used in TOOL code to separate items in a parameter list. Always include the commas in the parameter list.
colon :	Colons are used in TOOL code to separate a name from a type, or to indicate a Forte name in a SQL statement. Always include the colon in the type declaration or statement.
semicolon ;	Semicolons are used in TOOL code to end a TOOL statement. Always type a semicolon at the end of a statement.

Fusion Example Programs

Forte provides a number of example applications that illustrate Fusion features.

Process Client Example Programs

There are five different APIs available to build a Fusion process client. Example application programs are provided for each API. Each API has its own example files in a subdirectory of FORTE_ROOT/install/examples/conductr/. The PDF file in the examples subdirectory explains how to install and run the example application.

The examples are described in the appendix of the *Forte Fusion Process Development Guide*.

Backbone Example Programs

There are three Forte Fusion Backbone example programs: one illustrating the use of the Forte 4GL TOOL with Fusion, another is an example written in 'C', and a third example illustrates JMS messaging.

The example programs are installed under FORTE_ROOT/install/examples/fusion.

The directory containing each example includes a readme file. The readme file contains the background information and configuration instructions. The Forte 4GL example is a complete system, while the 'C' example is an optional replacement for some parts of the Forte 4GL example.

Documentation

The Fusion online documentation includes the complete documentation set and a master index as PDF (Portable Document Format) files as well as online help. For details on viewing and searching these files, see "Viewing and Searching PDF Files" on page 17.

When you are using a Fusion development application, press the Help key or use the Help menu to display online help. The help files are also available at the following location in your Fusion distribution: FORTE_ROOT/userapp/forte/cl*n*/*.hlp (*n* indicates the release number).

When you are using a script utility, such as Conductor Script (Cscript) or Fusion Script (FNscript), type help from the script shell for a description of all commands, or help <*command*> for help on a specific command.

Forte Fusion Documentation Resources

The *Forte Fusion Installation Guide* explains installation options and how to install the Fusion product (both the Fusion Process Engine and the Fusion Backbone).

Other useful resources available in the Fusion product documentation directory are:

- the fndoc.pdf file which serves as a home page for the entire documentation set
- a master index for all Fusion PDF documentation
- a glossary of terms
- a list of resources for learning more about the underlying technologies

Forte Fusion Process Management

The complete documentation set for Forte Fusion Process Management consists of the following manuals and online help:

- *Forte Fusion Process Development Guide*. Explains how to create business process logic using the graphical process development workshops.
- *Forte Fusion Process Management System Guide*. Explains system management concepts and facilities, how to register process definitions, how to configure and manage the process engine, and other related tasks.
- *Forte Fusion Process Client Programming Guide*. If you are building new applications that interact directly with the process engine, this manual explains how to use one of the provided process client APIs for that purpose. Use in conjunction with the API reference in the online help.
- Online help. Provides complete API reference for the process client APIs as well as task Help on the workshops and the Fusion Console.

Forte Fusion Backbone

The documentation set for the Forte Fusion Backbone consists of the following manuals and online help:

- *Forte Fusion Backbone System Guide*. Explains the backbone architecture, proxy concepts and features, and how to configure backbones and proxies. It includes a reference for FNscript, the Fusion scripting language. This should be the first manual you read if you are integrating an application or an adapter into a Fusion backbone.
- *Forte Fusion Backbone Integration Guide*. Explains how to develop XSL stylesheets and perform other integration tasks so that appropriate XML message transformations can occur between applications and proxies. The manual is used in conjunction with the *Forte Fusion Backbone System Guide* and the Fusion Backbone online help.
- Fusion Backbone online help. Explains proxy document XML, how to use the XML/XSL workshops to create, debug, and manage XML documents and XSL stylesheets, how XSLT and standard XML parsers are supported in Fusion, and provides a complete reference for the HTTPSupport (formerly the HTTP-DC) API. The Forte HTTPSupport API enables the development of standard HTTP communication services for transporting and managing HTML and XML messages.
- *Forte Fusion Adapter Development Guide*. Explains how to use the Forte Fusion software development kit to build a custom Fusion adapter for HTTP services. Discusses the design and functioning of a Fusion adapter, and includes guidance on using the 'C' and Forte 4GL TOOL adapter SDKs. Use in conjunction with the Fusion Backbone online help, which explains the HTTPSupport API, as well as with the adapter readme files. The following table indicates the location of readme files that contain further information about the 'C' adapter functions and the 'C' and TOOL adapter example programs:

Contents	Location
Installing TOOL adapter example program	FORTE_ROOT/install/examples/fusion/toolcon/readme.htm
Installing 'C' adapter example programs	FORTE_ROOT/install/examples/fusion/ccon/readme.htm
Explanation of 'C' adapter functions and #defines	FORTE_ROOT/install/fusion/ccon/fnconect.htm

Forte Application Environment

Forte provides a comprehensive documentation set describing the libraries, languages, workshops, and utilities of the Forte Application Environment. For the complete Forte Release 3 documentation set, see the Forte documentation listed on the Forte CyberSupport page at *http://www.forte.com/support*.

Viewing and Searching PDF Files

You can view and search Fusion PDF files directly from the product CD-ROM, store them locally on your computer, or store them on a server for multiuser network access.

There are two ways you can look up information in the Fusion documentation set:

- view and search PDF files directly from the product CD-ROM

  The Fusion documentation set has been indexed with Acrobat Catalog. Use Acrobat Reader with Search to search for text strings across the book set and click hypertext links to display the specified content.

- look up index entries in the *Forte Fusion Master Index* included on the product CD-ROM

  The master index also helps you find content across the full documentation set. It is a composite of all Fusion book indexes and is intended to be displayed online or printed to your local printer. It does not provide hypertext links to entries as the individual book indexes do.

Note You need Acrobat Reader 4.0+ to view and print the files. Acrobat Reader with Search is recommended and is available as a free download from http://www.adobe.com. If you do not use Acrobat Reader with Search, you can only view and print files; you cannot search across the collection of files.

▶ **To copy the documentation to a client or server:**

1. Copy the fortedoc directory and its contents from the CD-ROM to the client or server hard disk.

   You can specify any convenient location for the fortedoc directory; the location is not dependent on the Forte distribution.

2. Set up a directory structure that keeps the fndoc.pdf and the fusion directory in the same relative location.

   The directory structure must be preserved to use the Acrobat search feature.

Note To uninstall the documentation, delete the fortedoc directory.

▶ **To view and search the documentation:**

1. Open the file fndoc.pdf, located in the fortedoc directory.
2. Click the **Search** button at the bottom of the page or select **Edit** > **Search** > **Query**.
3. Enter the word or text string you are looking for in the Find Results Containing Text field of the Adobe Acrobat Search dialog box, and click **Search**.

   A Search Results window displays the documents that contain the desired text. If more than one document from the collection contains the desired text, they are ranked for relevancy.

Note For details on how to expand or limit a search query using wild-card characters and operators, see the Adobe Acrobat Help.

4. Click the document title with the highest relevance (usually the first one in the list or with a solid-filled icon) to display the document.

   All occurrences of the word or phrase on a page are highlighted.

5. Click the buttons on the Acrobat Reader toolbar or use shortcut keys to navigate through the search results, as shown in the following table:

Toolbar Button	Keyboard Command
Next Highlight	Ctrl+]
Previous Highlight	Ctrl+[
Next Document	Ctrl+Shift+]

6. To return to the fndoc.pdf file, click the Homepage bookmark at the top of the bookmarks list.
7. To revisit the query results, click the **Results** button at the bottom of the fndoc.pdf home page or select **Edit** > **Search** > **Results**.

Chapter 1

Introduction

The Forte Fusion Backbone is a suite of EAI tools based on XML and XSL technology that supports different styles of business integration across heterogeneous components.

This chapter describes at a high level the Forte Fusion Backbone architecture, explains the styles of business integration, and highlights typical integration tasks.

Forte Fusion and Business Integration

The Forte Fusion Backbone is a suite of business integration tools and software from the Forte Fusion product family.

The Forte Fusion Backbone is based on standard, configurable XML and XSL technology that enables rapid integration of applications into larger, integrated information systems that can span heterogeneous environments. These applications can be off-the-shelf software packages, including Web browsers, legacy applications, or new applications.

Forte Fusion enables immediate integration of any application with a native XML interface, using XML as the canonical format for message content, XSLT for message transformation, and HTTP or JMS for transport.

The Forte Fusion architecture supports the following styles of business integration:

- Point-to-point integration between applications, using direct XML messaging over HTTP and Fusion adapters. Fusion adapters are available to integrate many package applications, and the Fusion adapter toolkit enables integration of any custom application.
- Message brokering and XSL data transformation, using the Forte Fusion Backbone application proxies. XSL provides a standards-based way to share data across applications and domains.
- Using application proxies for message brokering and XSL data transformation but adding business process management. This solution uses all the capabilities of the Forte Fusion Backbone operating in concert with the Fusion process engine to provide end-to-end control and automation of all the processes in an enterprise’s business.

The key advantages of Fusion are:

- highly flexible, loosely-coupled business integration using XML and XSLT, supported by an XML/XSL workshop and stylesheet debugger
- multiple styles of business integration
- business process integration that can combine automated systems with human interactivity
- an enterprise-class process automation engine for support across multiple platforms, including OS/390
- a supporting software integration, development and runtime environment
- secure data transmission (SSL) for applications communicating over HTTP
- interface to the Java Message Service (JMS), which provides reliable messaging

Forte Fusion Backbone Architecture

A Forte Fusion Backbone application system consists primarily of an XML-based integration backbone. This backbone has the following components:

- A set of configurable application *proxies*, which perform message brokering and data transformation services on behalf of applications.
- A *Fusion backbone manager* to handle startup, shutdown, and general administration.
- An XSL *stylesheet base*, consisting of stylesheets in ASCII file format. Specific stylesheets from the stylesheet base are assigned to a proxy so it can interpret and transform the XML documents sent and received by partner applications.
- A configuration and administration tool, the Fusion Script (FNscript) utility, which provides the external interface to the backbone manager.

Where needed, a Fusion backbone system can also incorporate:

- Application *adapters*, used to exchange XML over HTTP with applications that do not present a native XML/HTTP interface.
- A *process engine,* provided by Forte Fusion, to support process-based integration. The Fusion process engine manages business processes using the set of process definitions created for the backbone.

Figure 2 shows an order processing system that incorporates all of these components in a process-based approach to integration.

Forte Fusion Backbone

A Forte Fusion backbone contains one or more application proxies that integrate separate components. Multiple backbones can operate in one Forte environment, each backbone functioning as an independent entity. The applications themselves are outside the backbone, and do not have to run in a Forte environment. Fusion backbones can coexist with Fusion process client applications in the execution of a business process (Fusion process clients interact directly with the same process engine.)

Each backbone has a configuration file that specifies details such as the name of the backbone, the backbone manager node, configuration details for each proxy, and whether the backbone uses a process engine or not.

Fusion Backbone Manager

The Fusion Backbone includes a backbone manager to handle change management of proxy configurations and restart proxies if they fail. There is one backbone manager per backbone.

Fusion Application Proxy

A Fusion application proxy is a configurable interface that brokers and transforms XML documents. The proxy can act as a client, server or client/server with respect to an application. It can also represent more than one application, if appropriate.

When used with the Fusion process engine, the proxy also acts as a client of the process engine, enabling the engine to command and respond to the proxy's partner application. This type of proxy is called a *process-based proxy.*

Because Fusion proxies are configured, not coded, you can quickly specify and modify how they function. The configuration information for a proxy includes all the information it needs to manage the expected XML messages and data as well as maintain sessions with its partner applications.

The proxy has the following key components:

- a configurable *XSLT processor*
- a *command processor*
- a *message broker*
- an interface to the application
- an interface to the Fusion process engine

These features are described in Chapter 2, "Proxy Concepts and Features."

For more details on backbone and proxy configuration, see Chapter 3, "Backbone System Configuration and Administration."

Forte Fusion Adapter

A Forte Fusion adapter is an optional component of a Fusion backbone. An adapter is required only when integrating an application that has no native XML/HTTP interface.

A Fusion adapter is partnered with the application proxy that represents its application in the Fusion backbone. If the application is a service provider, the adapter processes messages sent from the proxy. If the application is a service requestor, the adapter makes requests or posts messages to the proxy and receives responses.

A Fusion adapter serves the following functions:

- transforms XML documents sent by the proxy into requests invoked using the application's native API
- interprets the responses to these requests and generates appropriate application documents to be sent to the proxy
- responds to messages or application callback requests, generates appropriate documents, and forwards these to the proxy

For guidance on constructing a Fusion adapter, see the *Forte Fusion Adapter Development Guide*.

Styles of Business Integration

Fusion supports several integration options or styles, the most robust and scalable being process-based integration, and the simplest approach being point-to-point, using direct XML messaging over HTTP and Fusion adapters.

Process-Based Integration

The most robust and scalable style of business integration with Forte Fusion is process-based, taking advantage of the fault-tolerant Fusion process engine and supporting tools to automate and control all the business processes of an enterprise.

Business processes managed by Fusion can be complex, multi-step processes that involve:

- any type of business domain
- the exchange of many kinds of data
- transactions with high throughput
- multiple participating resources both human and mechanical
- timed activities
- complex subprocesses
- parallel streams of work
- activities that extend over long periods of time

In process-based integration, Fusion backbone proxies perform message brokering and data transformation on behalf of applications that participate in business processes. The process engine maintains activity state information for the proxy and provides recovery and failover.

When the process engine is used to manage a business process, process logic is independent of application logic, making it easy to implement dynamic changes in the business process. The use of XSL to specify the actions involved in executing the business process enables integration logic to be developed and maintained separately as well.

A typical example of a complex, end-to-end business process is order fulfillment, in which customers place orders on the Web and the orders are sequentially routed to and processed by other applications that perform discrete functions to fulfill the order, such as credit verification, inventory management, shipping and billing.

Figure 2 shows an order processing system involving three coordinated applications. The business process is managed by a Fusion process engine. Two applications use a Fusion adapter to communicate with a partner proxy in the backbone; the other has a native XML interface and interacts directly with its partner proxy. Each application can have its own XML schema or dialect. All possible components of a Forte Fusion backbone system are used in this sample configuration:

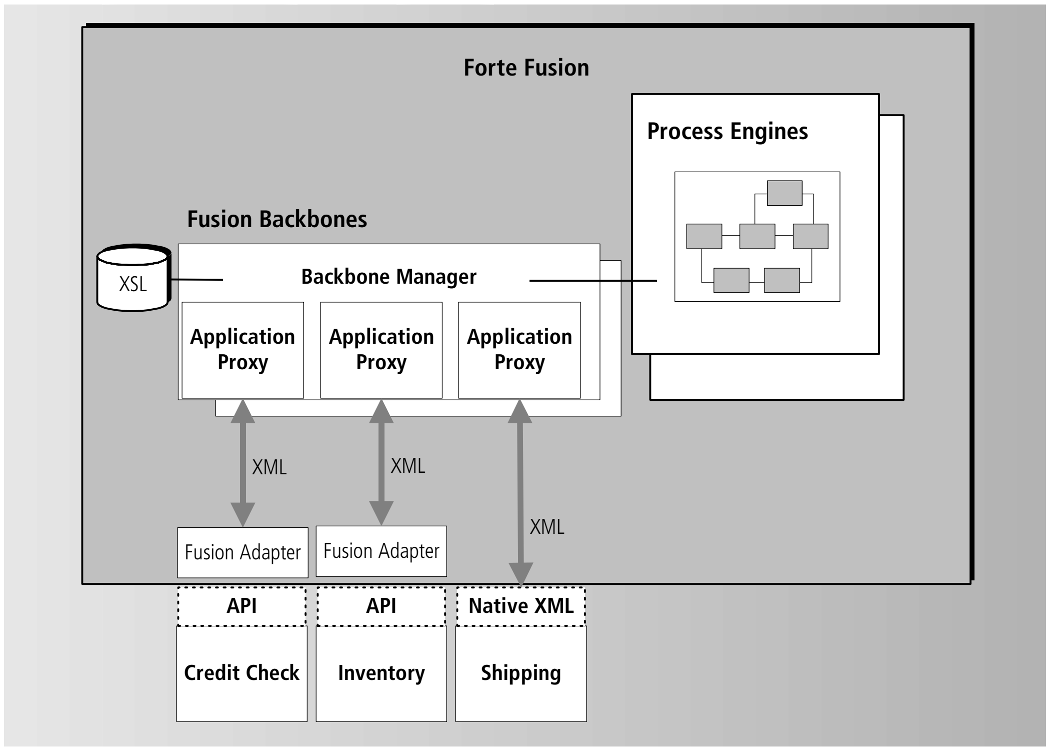

Figure 2 *A Fusion Order Processing Backbone*

Simple Integration

A simple approach to integration is to implement a Fusion backbone consisting of one or more application proxies, and supply the proxy with appropriate XSL stylesheets. Simple integrations can be appropriate for small projects and prototypes, and can also be migrated upwards as business needs dictate. All components of a Fusion system can come into play in this style of integration, except for the Fusion process engine.

Figure 3 shows three application proxies that broker and transform messages exchanged between three partner applications. One application has a native XML interface, the others use a Fusion adapter. Each application can use its own XML schema or dialect.

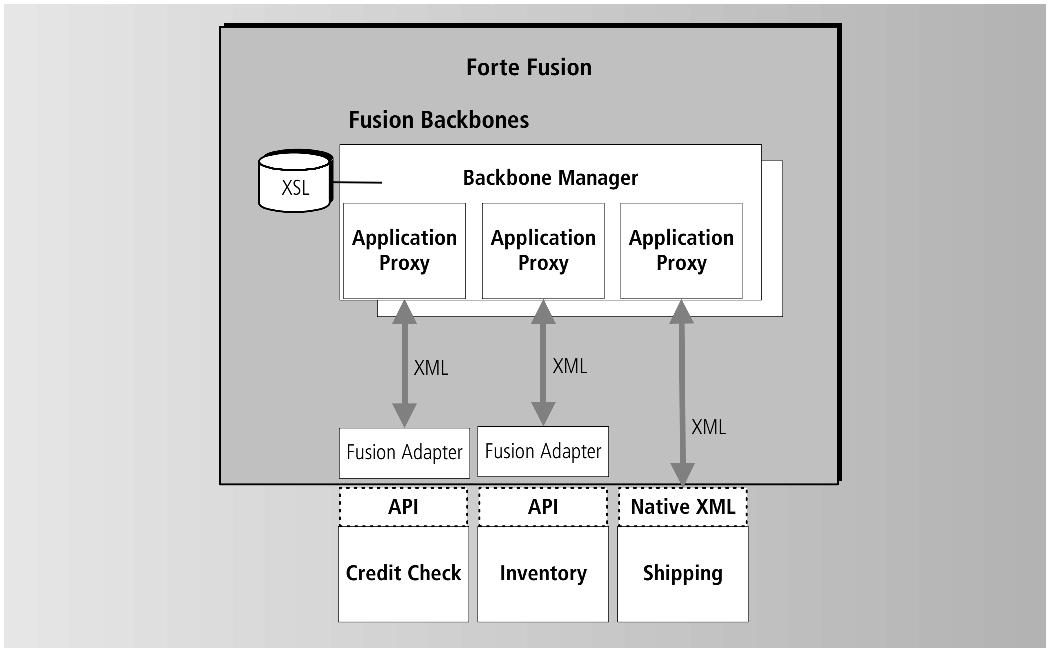

Figure 3 *Integration for XSL Transformation*

Note Proxies that do not interact with a process engine, called *independent proxies*, still require the installation of the Fusion process engine runtime. They must also be explicitly configured to operate without a Fusion process engine. For details on configuring proxies appropriately, see Chapter 3, "Backbone System Configuration and Administration."

Integration Tasks

This section outlines basic tasks involved in integrating applications into a Fusion backbone application system. References to detailed documentation are included at the end of the section.

- Determine which applications you want to integrate.
- Determine whether process-based integration is appropriate for your needs. Simpler styles of integration can be appropriate for smaller projects and prototyping; such implementations can be migrated upwards to meet changing business needs. Process-based integration is the most robust and scalable approach for enterprise integration.
- For point-to-point integration, analyze the data requirements of the applications. Use Fusion adapter technology and services to provide direct XML message translation.
- For process-based integration:
  - Decide on the functional grouping of applications according to business processes. Multiple processes can be represented to the Fusion process engine by one backbone.
  - Determine how each application interacts with the business process. An application is either a service requestor (such as a Web client), or a service provider (such as a credit verification service). An application can be both a provider and a requestor. It can interact with a human user or perform its work automatically.
  - Develop and register the process sequencing logic using the Fusion process definition workshops.
- For any integration that uses proxies for message brokering and transformation:
  - Analyze the data requirements for the application and develop your use cases. For process-based integration, these use cases include interactions with the process engine. When the proxy is used independently, the use cases represent interactions between applications.
  - Develop sample XML documents in the XML vocabulary of your application to represent each application's communication with the proxy for a given use case.
  - Write the XSL stylesheets the proxy applies to inbound and outbound documents. Refer to the Fusion proxy document vocabulary as appropriate.
  - If you configure a proxy to use HTTP and an application has no native XML/HTTP interface, use a Fusion adapter or build one (using the adapter toolkit) to provide HTTP services and XML parsing.
  - Configure backbones and their proxies for appropriate interactions with partner applications and optionally with the process engine.
  - Determine how XML documents are to be routed between applications.

For details on:

- Developing process logic, see the *Forte Fusion Process Development Guide*.
- Proxy concepts and features, see Chapter 2, "Proxy Concepts and Features."
- The XML documents processed by the proxy, see Chapter 4, "Proxy Documents" and the Fusion Backbone online help.
- Creating, editing, debugging and managing collections of sample XML documents and XSL stylesheets using the XML/XSL Workshops, see the Fusion Backbone online help.
- Use cases and Fusion XSL stylesheets, see the *Forte Fusion Backbone Integration Guide*.
- Configuring backbones and proxies, see Chapter 3, "Backbone System Configuration and Administration."
- Building a Fusion adapter, see the *Forte Fusion Adapter Development Guide* and the HTTPSupport API reference in the Fusion Backbone online help.

Chapter 2

Proxy Concepts and Features

The proxy is the key component of a Fusion backbone. Its primary role is to broker and transform XML documents exchanged between the integrated application or applications that it represents. When applications participate in a business process, it also provides the XML interface between the Fusion process engine and the application.

This chapter explains the internal components of the proxy and their functions, and the proxy's capabilities with respect to managing sessions and ensuring reliable operation. For details on how to configure a proxy, see Chapter 3, "Backbone System Configuration and Administration."

Proxy Components

A Forte Fusion application proxy is a configurable interface that exchanges and transforms XML documents flowing between backbone-integrated applications.

The proxy has the following key components:

- a configurable *XSLT processor*, to transform inbound and outbound XML documents
- a *command processor*, to generate command-related information
- a *message broker*, to send and receive XML documents containing commands or other information
- an interface to the application
- an optional interface to the Fusion process engine

This manual refers to a proxy that interacts with a Fusion process engine as a *process-based proxy*, to distinguish it from an *independent proxy*.

- Process-based proxies on a Fusion backbone represent applications that participate in a business process managed by a Fusion process engine. The business process is defined in the Fusion process development workshops and registered with the process engine. The proxy performs message broker and data transformation functions on behalf of backbone-integrated applications that are using XML to exchange instructions and data relevant to executing the business process or for other purposes.
- Independent proxies on a Fusion backbone do not interact with a Fusion process engine. The proxy only performs message broker and data transformation functions on behalf of applications that use different dialects of XML to exchange information.

Note A process-based proxy can route and transform other messages unrelated to the business process without using the process engine. A process-based Fusion backbone can also include one or more independent proxies. For details, see “Fusion Process Engine Usage.”

XSLT Processor

When a proxy is used for data transformation, you supply it with XSL stylesheets that transform the XML documents it exchanges with applications. The proxy’s XSLT processor applies these stylesheets to convert source documents into result documents. Result documents contain commands to be performed, such as sending messages to other applications.

If the proxy is process-based, one of its primary functions is to send commands to the Fusion process engine, for example, to create a new instance of a business process. In process-based integration, XML documents flow in two directions through the proxy, either originating in the application (or adapter) and resulting in Fusion process engine commands being invoked by the proxy, or originating as Fusion process engine events. The engine publishes the events to the proxy and the proxy forwards appropriate XML documents to the application.

The XML documents processed by the Fusion application proxy are called *proxy documents*. The proxy documents that originate from and are sent to an application are defined in the application's XML vocabulary; this type of proxy document is called an *application document*. Two other proxy documents (called *command documents* and *state documents*) are used internally within the proxy and have a vocabulary and structure known to the proxy. Command documents contain instructions for the application or the Fusion process engine; state documents convey change of state information and originate from or are sent to the Fusion process engine. A fourth type of proxy document, called *authentication document*, is used for HTTP session authentication. For an overview of the proxy documents defined in Fusion, see Chapter 4, "Proxy Documents."

To understand the XSL stylesheets appropriate for Fusion, see the *Forte Fusion Backbone Integration Guide*.

Command Processor

The list of commands to be performed based on a proxy document is itself an XML document, and is called a *command document*. The proxy has a *command processor* to interpret these command documents and generate messages regarding the commands to be performed by the process engine and the application.

Message Broker

The proxy's message broker sends and receives XML documents. For example, it can route proxy documents to the application or the Fusion process engine, as well as send and receive messages exchanged between applications. These messages can contain instructions and data needed to execute a business process, or for other purposes.

Application Interface

An application communicates with a partner application proxy using XML sent over HTTP or with messages sent using the Java Messaging Service (JMS). Communication is typically a one-to-one relationship, but there can be multiple applications for one proxy, or multiple proxies for one application.

In a process-based Fusion backbone, an application can either request or provide services within the business process. Using HTTP terminology, to a service provider application, the proxy presents itself as a client; to a service requestor application the proxy presents itself as a server or client/server. A process-based proxy can act in any of these capacities; an independent proxy can only be a service provider.

- A proxy acting as an HTTP server or JMS listener accepts XML documents as the message content and performs appropriate actions based on the content. A process-based proxy, for example, might request the process engine to initiate a business process or complete a business activity; an independent proxy might forward a request.
- A proxy acting as an HTTP client or JMS sender creates XML documents that are forwarded to the application. The proxy sends these documents during proxy initialization (for in-progress work) and when it accepts new work from the process engine and the integrated applications. (See the following section, "Fusion Process Engine Interface".)

Note In JMS terminology, a *MessageProducer* is an object used for sending messages to a destination and a *MessageConsumer* is an object used for receiving messages sent to a destination. This manual refers to MessageProducers as *JMS senders* and MessageConsumers as *JMS listeners.*

For examples of typical interactions between the engine and the application according to the proxy's function as an HTTP client or server (or JMS sender/listener), see the *Forte Fusion Backbone Integration Guide*.

Fusion Process Engine Interface

A Fusion process engine interface is only applicable for process-based proxies. (You must configure the proxy or backbone to use this interface or not; for details, see "Fusion Process Engine Usage" on page 42.)

To communicate with the Fusion process engine, a proxy needs to know the process engine session information for its application, which includes roles and user profile, as well as how to handle recovery and session authentication.

During initialization the proxy opens a session with the Fusion process engine. The newly initialized proxy obtains its list of in-progress (ACTIVE) work activities from the process engine's persistent work queue manager, and sends an appropriate message to the application for each work activity. The proxy then waits for new work events from the engine and new messages from the application.

The primary tasks of the Fusion engine interface are to invoke Fusion process client API commands sent by the application, and to handle events passed from the engine to the proxy. The engine interface automatically interacts with the process engine to change newly READY activities to ACTIVE and translate related information into messages that are sent to the proxy. The proxy relies on the process engine to maintain a list of ACTIVE activities.

The Fusion process engine interface directly handles some events, such as events that report the removal of an activity from the activity list, as well as state changes from READY to ACTIVE, without involving other elements of the proxy. It also responds to instructions received from the application to tell the process engine that ACTIVE work is either completed or aborted. For details on activity states, see the *Forte Fusion Process Development Guide*.

Session Management

The proxy can act as a service requestor or service provider (or both) with respect to an integrated application. A service requestor application requires a service provider proxy; a service provider application requires a service requestor proxy. An independent proxy can only act as a service provider. A process-based proxy's behavior corresponds to the way the application participates in the business process.

When establishing a session with an integrated application, a proxy communicates with the application by sending and receiving messages containing XML documents (or other information). The proxy communicates with the application using either HTTP or the Java Messaging Service (JMS).

The proxy's configuration includes all the information it needs to maintain sessions with the application and with the Fusion process engine, as appropriate.

Fusion Process Engine Sessions

A process-based proxy is a client program with respect to the Fusion process engine. It makes direct API calls to the engine to open and close the session with the engine and make process-related requests. It also processes event-related information sent to it by the process engine. The engine communicates directly with proxies, but not with the applications the proxies represent.

A process-based proxy always establishes the engine session when it starts. The engine requires the Fusion user profile to validate the user session. The Fusion user profile is defined in the User Profile Workshop and registered with the engine (as explained in the *Forte Fusion Process Development Guide*). The user profile information supplied by the proxy includes user profile name, user name, password and role(s). This information must be configured for the proxy; the values are stored in an encoded form in the configuration file.

HTTP Sessions

Service requestor applications establish an HTTP session with the proxy and send it XML/HTTP requests. The proxy acts as an HTTP server or client/server, handling requests and sending back responses.

When the proxy interacts with a service provider application (a scenario only appropriate in process-based integration), the proxy acts as an HTTP client. It first establishes a Fusion engine session to initiate a request, and proceeds to open an HTTP session if appropriate.

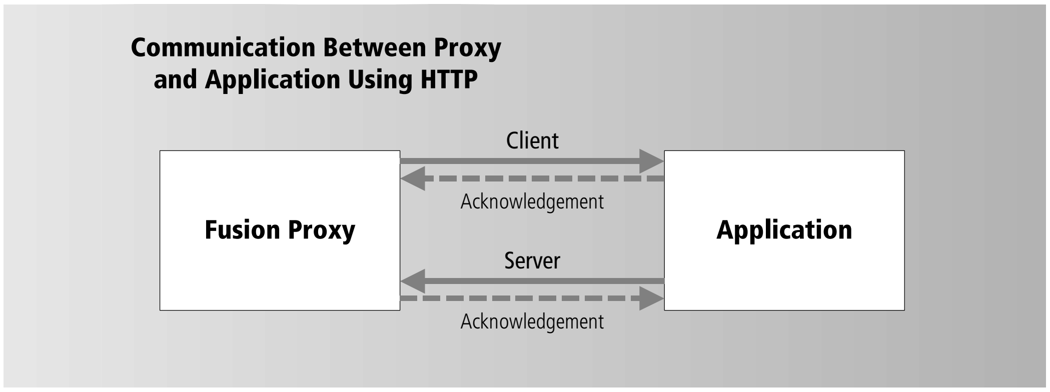

Figure 4 *Communication Between Fusion Proxy and Application Using HTTP*

A proxy must be configured appropriately as an HTTP client, server, or client/server. For details, see Chapter 3, "Backbone System Configuration and Administration."

Synchronous and Asynchronous Processing Using HTTP

HTTP is a synchronous protocol; there is always a response to a request. However, the response does not necessarily provide the information needed to handle the request; other processing may take place in the interim, and it may take time for the request to be fulfilled.

Synchronous mode

When the proxy represents a service provider, the interaction is either synchronous or asynchronous with respect to work done by the application. In either case the operation begins when the Fusion engine passes a READY activity event to the proxy. The proxy automatically advances the state of that activity to ACTIVE. The proxy sends an application document to the application that contains appropriate instructions and data.

If the proxy and the application interact synchronously, the proxy does not take any action until the application notifies it that the activity has been completed (or aborted or rolled back). The proxy assumes that the application's response refers to the current process or activity and it acts upon that assumption. The application does not need to pass the process ID and activity ID back to the proxy.

Asynchronous mode

If the proxy and application are interacting asynchronously, the proxy may perform other actions before the application notifies it about the status of the activity. In this scenario:

- the application must supply the process ID and activity ID
- the XSL stylesheet that covers the use case must ensure that these IDs are made available
- the proxy must be configured for client/server communications with the application

For details on Fusion use cases (synchronous and asynchronous) and constructing appropriate XSL stylesheets, see the *Forte Fusion Backbone Integration Guide*.

Using SSL

Support for SSL is built into the Fusion proxy and can be turned on for an HTTP server or HTTP client proxy so that it can negotiate the SSL handshake protocol with the application. When you enable SSL (using **SetSSL**), you can also specify the levels of secrecy, integrity, authentication, and compression, as well as whether connections should be automatically resumed, and the usage of certificates.

The SSL 3.0 specification stipulates that authentication with certificates is optional on the client side, but required on the server side. If the proxy is an HTTP client, you can configure it only for secrecy and integrity or you can include authentication. If the proxy is an HTTP server and SSL/HTTPS is required for the connection with the application, the backbone must have certificates registered for it (using **RegisterCertificate**) and you specify which certificates authenticate the proxy along with the other SSL options (using **SetSSL**).

For the procedure on registering certificates, see "Registering SSL Certificates" on page 49; for details on configuring SSL properties, see "Configuring HTTP Proxies" on page 54.

HTTP Session Authentication

HTTP session authentication allows the proxy to require or provide authentication to interact with an application. With this feature enabled, only authorized users are able to obtain access. The request/response messages contain authorization headers that are managed automatically by cookies.

A process-based proxy can support HTTP session authentication by referring to an application user profile and application role. A service requestor application can require the proxy to provide authentication. The proxy provides this authentication with the help of the process engine. Similarly, an HTTP client proxy can require the application to provide session authentication.

An independent proxy can also be configured for HTTP session authentication by supplying it with credentials it can verify on a local machine.

Authentication schemes

The available authentication schemes are Basic and FusionXML. Both types of authentication strings are encoded in Base64 before transmission:

- Basic, as defined in HTPP 1.1, consists of a *name:password* combination that is returned in the authorization header.
- FusionXML, an extension defined by Fusion, allows more complex user data to be transmitted, such as that contained in a user profile. FusionXML is a string that is returned in the WWW-Authenticate header that lets the application or the proxy know the authentication scheme that is required. For a description of the authentication document vocabulary and structure, see the Fusion Backbone online help.

Note

Only Basic authentication is supported for independent proxies, and the credentials must be provided on the local machine. Such proxies only act as HTTP servers and service providers. For details, see "HTTP Server Proxy Authentication" on page 34.

Authorization header

When an HTTP client (either the proxy or its application) makes a request to an HTTP server, session authentication proceeds as follows:

1. The server determines whether session security is required.
2. If so, and a session does not exist, the server returns the request with a status code of Unauthorized (401) along with a specification of the authentication scheme (Basic or FusionXML) that is required.
3. When the client receives this information, it resends the original request with a completed authorization header that uses the required authentication scheme.
4. When the server receives the required information, it validates the session request and takes appropriate action, typically fulfilling the original request. If the information it receives is incorrect, it rejects the attempt by sending another Unauthorized response.

HTTP Client Proxy Authentication

If the service provider application requires session authentication, the proxy fulfills the authentication scheme request and returns the Authorization header with the completed user information.

You configure the client proxy for session authentication using the following FNscript commands:

- **SetAplSession** to specify a user name, password, and an “otherInfo” string
- **AddAplRole** to specify an application role (for example, OrderEntry)
- **SetAplProfile** to specify a user profile name

For details on specifying session authentication for a client proxy, see “Client proxy session authentication” on page 54.

HTTP Server Proxy Authentication

To specify that a server proxy requires authentication of service requestors, you configure the proxy’s authentication scheme to either Basic or FusionXML. If a server proxy is not configured with an authentication scheme, it does not require authentication of service requestors.

When a service requestor application requests that a process instance be started, the following steps occur:

1. The proxy determines whether session security is required.
2. If so, and a session does not exist, the proxy returns the request with a status code of Unauthorized (401) along with a specification of the authentication scheme (Basic or FusionXML) that is to be used when the service requestor responds with the Authorization header.
3. When the service requestor receives this information, it resends the original request, along with a completed Authorization header that uses the required authentication scheme.
4. When the proxy receives the required information, it validates the session request and takes appropriate action, typically fulfilling the original request. If the information it receives is incorrect, it rejects the attempt by sending another Unauthorized response.

You configure the server proxy for session authentication using the following FNscript commands:

- **SetAuthentication** to specify either Basic (*username:password* only) or FusionXML and whether the authentication credentials are supplied by the process engine or locally. Independent proxies can only use local authentication and the Basic scheme.
- **SetCredentials** to specify local credentials for independent proxies. The proxy compares these credentials, encoded in the configuration file, with the user name and password supplied by the application.
- **SetSessionTimeout** to specify how many seconds until a server session times out.
- **SetProtocol** to tailor the use of session authentication if needed.

Note You can specify the maximum number of sessions using the **SetSessionMaximum** command. For information on specifying session management properties, see “Setting Recovery and Reliability Options” on page 52.

If the proxy is acting as both a client and server, you need to configure it using both sets of FNscript commands. For information on specifying session authentication for a server (or client/server) proxy, see “Server proxy session authentication” on page 55.

JMS Sessions

Java Message Service (JMS) is a specification that allows separate applications to communicate with each other asynchronously, using objects created and maintained by a JMS provider application (such as JMQ). These administered objects are made available on a network by the Java Naming and Directory Interface (JNDI).

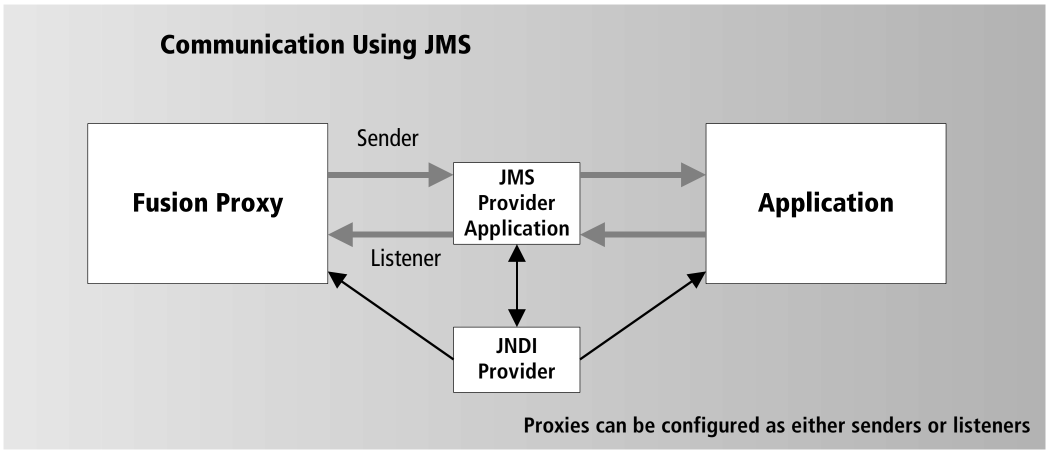

Figure 5 *Communication Between Fusion Proxy and Application Using JMS*

In a Fusion Backbone, proxies can be configured to use JMS when communicating with integrated applications. You can use the **SetProtocol** FNscript command to specify configuration information for proxies that use JMS. The **SetProtocol** command is available at both the backbone and proxy levels.

Use the **AddAplURL** command to specify the proxy as a JMS sender. Use the **AddListener** command to specify the proxy as a JMS listener. These commands are only available at the proxy level.

Note This section provides a brief synopsis of JMS to introduce you to the Fusion Backbone interface to this service. For detailed information on JMS, refer to the Java Message Service specification, available at http://java.sun.com/products/jms/docs.html. For the JMS providers supported by this release, refer to the platform matrix at http://www.forte.com/support/platforms.html.

JMS Messaging Models

There are two types of messaging models you can use with JMS, *publish/subscribe* and *point-to-point*. In the publish/subscribe model, a JMS sender publishes a message to a *topic*—multiple JMS listeners can subscribe to published topics. In the point-to-point model, a message has at most one recipient. The JMS sender addresses the message to a *queue* that holds the messages. Each message can be taken by only one listener. Topics and queues are objects administered by the JMS provider application, using JNDI.

JMS Durability

JMS provides for durable subscriptions to published topics. JMS retains the state of durable subscriptions, allowing subscribers to resume access to a topic after interruptions in service. Durable subscriptions incur more overhead than non-durable subscriptions, which last only for the lifetime of the subscriber.

JMS Message Selectors

JMS message selectors allows a JMS listener to specify the messages it is interested in receiving. Only messages that match selector properties specified by the listener are delivered. When sending a message, a JMS sender can specify *message properties* that the listener can choose to use for message selection.

Message properties are *name/value* pairs, where *name* is a valid JMS property name and *value* is a string that can be converted to the Java type of the property. A JMS listener can register with a queue or topic for specific message properties, and thus filter the messages it receives.

The following property types are supported: Boolean, Byte, Double, Float, Int, Long, Short, and String. (Object property types are not supported.)

For a proxy configured as a sender, specify the properties of outgoing messages with the **AddAplURL** FNscript command. For a proxy configured as a listener, use the **AddListener** command to specify which properties the proxy is interested in receiving.

JMS Authentication

As mentioned previously, access to administered objects is provided by a JMS provider application using JNDI. Authentication and security issues are handled by the JMS provider application.

Fusion supports the use of JNDI simple authentication or JNDI with no authentication at all. In simple authentication, you provide a user/password to the JNDI utility. Use the **SetProtocol** FNscript command at the backbone or proxy level to supply a user name and password for JNDI authentication.

You can also use the **SetAplSession** command to supply a user name and password for JMS authentication at the proxy level (for both listener and sender sessions).

JMS Reliability

JMS addresses the problem of reliable data transmission across networks by offering a “once-and-only-once” delivery system. Messages are sent to a topic or queue, and listeners can specify acknowledgement according to how you balance the requirements of reliability and performance.

A listener can be configured for *client acknowledgement* or *automatic acknowledgement*. When configured for client acknowledgment, the proxy (after it finishes its work) sends a message acknowledgement to the JMS provider. If configured for automatic acknowledgement, the JMS provider assumes the work has been complete as soon as it delivers a message (no acknowledgement is sent by the proxy).

Use the **AddListener** command to specify the acknowledgement mode when configuring a proxy as a listener.

Note Forte Fusion does not support the transactional receipt of JMS messages.

Recovery and Reliability

The Fusion proxy supports a number of features for load balancing, recovery and reliability:

- For better management of proxy resources, you can specify the maximum number of activities a proxy can start at one time, as well as set timeout values. For example, you can prevent starting more concurrent activities than an application can manage.
- You can set parameters for retrying communication with an application.
- You can specify alternative partner applications for a proxy to contact in the event of an application failure.
- The default configuration for proxies sets them to start automatically when a backbone starts, as well as restart after a failure.
- For better control of a proxy's behavior before and after shutdown, you can specify various shutdown options, for example, to cause the proxy to shut down only when there are no more ACTIVE-state activities, and to restart immediately after shutdown.
- For application load balancing, the proxy can access the application at multiple locations.
- You can ensure that an application is always available to its proxy.

For details, see Chapter 3, "Backbone System Configuration and Administration."

Chapter 3

Backbone System Configuration and Administration

A Forte Fusion backbone configuration consists of information needed by the backbone and its proxies to interact with the rest of the Fusion application system.

This chapter provides guidelines for planning a backbone, and explains how to configure a backbone and proxies.

This chapter also discusses system administration tasks.

Planning a Fusion Backbone

A Fusion backbone system runs in a Forte software environment, using Fusion development and production capabilities. A functioning Forte environment, Fusion process engine runtime, and Fusion Backbone runtime components together provide the technical infrastructure. (The Fusion process engine runtime is required even if the backbone does not interact with a process engine.)

To set up a new Fusion backbone system, install the Fusion Backbone system software in an existing Forte and Fusion process management environment. For details on installation, see the *Forte Fusion Installation Guide*. You can then configure one or more backbones.

The Fusion application runtime software consists of the following partitions and libraries:

- Fusion backbone manager—a Forte server partition
- Fusion application proxy—a Forte server partition
- Fusion Script (FNscript) utility—a Forte client partition
- XML/XSL Workshops—libraries that provide a graphical client tool for creating and debugging XSL stylesheets
- TOOL Adapter SDK—libraries that are part of Forte and enabled during the Fusion Backbone installation
- C Adapter SDK—source code libraries

Configuration Options

A Forte Fusion backbone configuration consists of a backbone manager, one or more application proxies, an XSL stylesheet base, the server node identification for the backbone manager and each proxy, and information needed by the proxy to interact with the rest of the Fusion application system. If any proxies on the backbone are process-based, the configuration includes the name of the Fusion process engine the backbone will use.

Partitioning

You have many options when planning a backbone system; the components and settings you choose depend on your environment and its capabilities and the requirements of the business process. For example, a backbone can interact with a process engine or not, and Fusion adapters are only needed for HTTP-based applications that do not have native XML interfaces. A backbone could consist of only one application proxy and the backbone manager.

Consider the following options when planning the partitioning of your system:

- You can configure multiple Fusion backbones in one Forte environment. Each backbone functions as an independent entity.
- Each backbone can have multiple proxies. If the backbone is process-based, it can still have proxies that do not interact with a Fusion process engine, if appropriate.
- The backbone manager and its proxies do not have to be on the same node. (You must add the backbone manager and proxy node names to your Forte environment.)
- For process-based integration, backbone components do not have to reside on the same node as the components of the Fusion engine that supports the backbone.
- A single Fusion process engine can host multiple backbones.

Figure 6 shows all the possible components of a Fusion backbone system. This sample system has two backbones, each with its own stylesheet base, and two process engines. (In another system, one engine might host multiple backbones.) One application communicates directly with its partner proxy, the other two use Fusion adapters to communicate with the proxy.

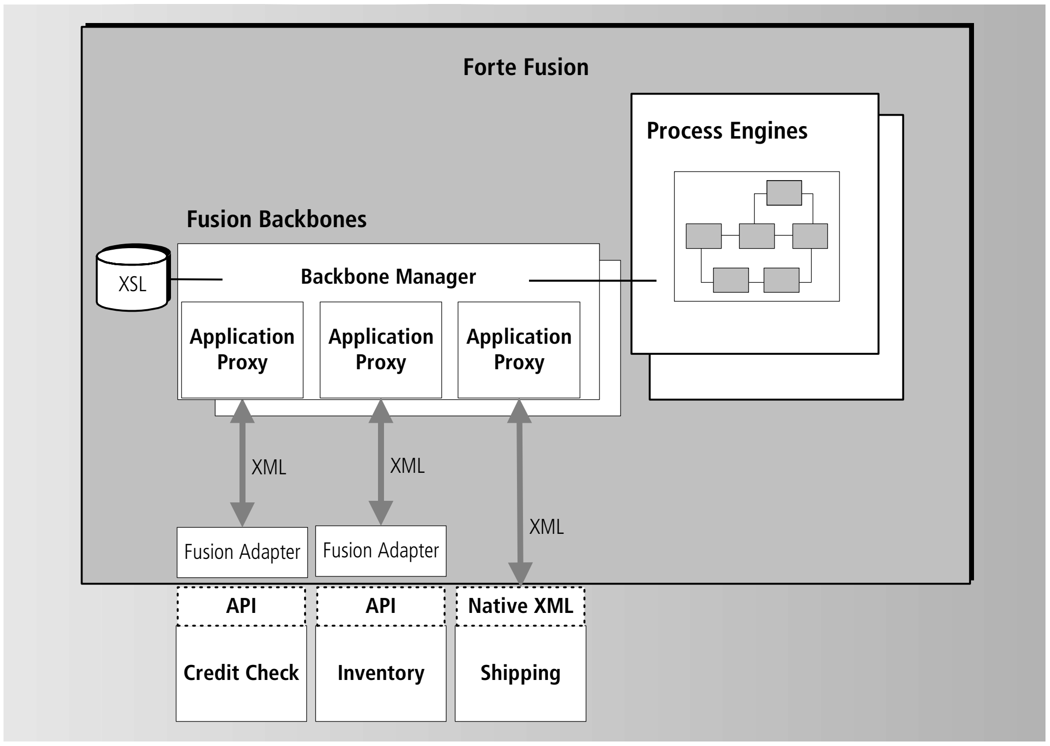

Figure 6 *A Sample Forte Fusion Backbone System*

Message Brokering

Proxies in a Fusion backbone can communicate with partnered applications using XML messaging. This communication can be accomplished using either the HTTP protocol or the Java Messaging Service (working with a JMS provider application). A Fusion backbone can contain a combination of proxies that use either HTTP or JMS.

Communication over HTTP allows a proxy to establish either synchronous or asynchronous sessions with service providers and can also implement security features available using Secure Sockets Layer (SSL). HTTP provides a number of advantages for ease of integration:

- Many applications accept HTTP/XML messages in a published format.
- When writing adapters, you have a wide choice of programming languages.
- When writing adapters, HTTP client and server code is often freely available.
- HTTP does not rely on another product for delivering messages.

For situations requiring guaranteed message delivery, you can take advantage of the “once-and-only-once” delivery feature available with JMS. Communication using JMS is inherently asynchronous.

Reliability, Scalability and Security

The proxy can be configured with various options for improving the reliability and scalability of your system.

- For scalability, you can configure Fusion services to operate on different machines.
- For better management of proxy resources, you can specify the maximum number of activities a proxy can start at one time.
- For reliability, you can set retry parameters as well as provide alternative partner applications for a proxy to contact in the event of an application failure. The default configuration for proxies sets them to start automatically when a backbone starts, as well as restart after a failure.
- An HTTP proxy can be configured to require or provide session authentication to its partner application as well as to use SSL.

For details on recovery and reliability settings, see “Setting Recovery and Reliability Options” on page 52. For details on session authentication and SSL, see “Configuring HTTP Proxies” on page 54.

Adapters and Applications

Where adapters are used, they represent the application to the proxy and therefore have some properties, such as URL, that might need to be indicated in the proxy’s configuration. If an adapter is used, the adapter becomes the proxy’s partner application. Properties such as “application URL” are therefore properties of the adapter, not the application it represents.

Fusion Process Engine Usage

By default, backbones and proxies are configured to operate with a Fusion process engine. You can change the engine usage property (**UseProcessEngine**) at the backbone and the component level of FNscript, as appropriate. For details on these levels, see Appendix A, “Fusion Script Commands.” A backbone that operates with a process engine can also have member proxies that do not.

- If a backbone is process-based, name the engine to be used.
- If a backbone does not use a process engine at all, specify at the backbone level of FNscript that the backbone operates without an engine.
- If a backbone is mainly process-based but one or more proxies on it operate without an engine, specify at the component level of FNscript (the proxy is current when you issue the command) that the proxy operates without an engine. This setting does not affect the rest of the proxies on the backbone.
- For a backbone that has both process-based and independent proxies, you can configure the backbone in the way that applies to most of the proxies, and set the other proxies’ engine property on or off. Name the process engine to be used by the process-based proxies on the backbone.

You use the following FNscript commands for the above functions:

- **SetCEngine** *engine_name* to name the Fusion process engine used with the backbone.
- **UseProcessEngine off** to turn off the default usage of the process engine for the backbone (issued from the backbone level of FNscript).
- **UseProcessEngine off** to turn off the default usage of the process engine for a specific proxy (issued from the component level of FNscript).

Note Independent proxies, because they operate without the Fusion process engine, are unable to maintain state information and important data may therefore be lost. Process-based integration is the most reliable integration solution.

Backbone Configuration

For the backbone and its manager you must configure:

- the backbone name
- backbone manager node (as specified in your Forte environment)
- Fusion engine usage (on or off)
- Fusion engine name, if the backbone is process-based
- XSL stylesheet base
- SSL certificates, if any backbone proxies use SSL for authentication

XSL stylesheets

When you register a stylesheet for the backbone, a persistent copy of the stylesheet is created in the backbone's stylesheet base. The stylesheet base is maintained on the Forte central server node. You can register stylesheets directly from the XML/XSL Plan Workshop, or use the FNscript command **RegisterRule**.

You configure proxies to use specific stylesheets from the stylesheet base, as described in the next section. Stylesheets can be shared by proxies on the same backbone, but not between backbones.

SSL certificates

Like stylesheets, certificates are maintained on the Forte central server node and managed at the backbone level. You can register, deregister and list currently registered certificates. You cannot use SSL for authentication on a specific proxy-application connection unless you have registered certificates with the backbone to which the proxy belongs and configured the proxy for SSL usage, as described in the next section. Certificates can be shared by proxies on the same backbone, but not between backbones.

A file containing a single certificate identifies an individual server or client (along with a password). A file containing the certificates of one or more acceptable roots is used for accepting authentication information. The filenames of the files are encrypted; you use passwords to decode them for use in authenticating.

Proxy Configuration

Each proxy must be configured with:

- proxy name (unique within the backbone)
- proxy node (as specified in your Forte environment)
- XSL stylesheets (inbound and outbound)
- application session information
- Fusion engine usage and session information (for process-based proxies)
- desired recovery and reliability properties
- other desired features, such as HTTP session authentication and SSL

XSL Stylesheet Settings

The XSL stylesheets you apply to the proxy must already be registered with the backbone, either from the XML/XSL Workshop or through FNscript. You need to specify the direction in which each stylesheet applies:

- Outbound stylesheets apply to documents that flow from the proxy to the partner application.
- Inbound stylesheets apply to documents that flow into the proxy from the partner application.

The proxy's XSL stylesheet configuration names the stylesheet for each direction. If a stylesheet is not configured for direction (inbound or outbound), XML documents are routed to their destinations without being processed.

Configuring Proxies as HTTP Clients and HTTP Servers

Proxies can communicate with their partner applications by means of HyperText Transport Protocol (HTTP). Every HTTP exchange is a two-way communication, in which a client sends an initial request, and a server must send a response to complete the request.

Thus, each proxy must be configured appropriately as a client or server, or sometimes both, depending on how it communicates with its partner application. Table 1 shows the possible proxy roles and the FNscript command to use for each:

Table 1 *Configuring Proxies as HTTP Clients and HTTP Servers*

Proxy Role	FNscript Command	Information Required

HTTP server—the proxy receives requests from a client application.	**SetPort** *port_number*
	AddListener
	Protocol=HTTP
	Port=*port_number*
		The port number to which the application should send requests. For example, to configure a proxy to receive requests at port 5010, issue the following FNscript command:
		fnscript> SetPort 5010

HTTP client—synchronous communication; the proxy submits a request to an application and receives a completion response during the same HTTP session. For details on asynchronous processing, see "Synchronous and Asynchronous Processing Using HTTP" on page 32.	**AddAplURL** *url_name*	A persistent URL and port number for the application. For example, to send requests to an application at *http:sera.abbysys.com:555*, issue the following FNscript command: fnscript> AddAplURL http:sera.abbysys.com:555 You can issue this command multiple times to configure a proxy with additional URLs. The first one in the list is by default the primary partner; you can use the additional locations for load balancing or failover (for details on accomplishing this task, see "Setting Recovery and Reliability Options" on page 52).
HTTP client and server—asynchronous communication; the proxy acts as an HTTP client to submit requests for work to an application; the proxy acts as an HTTP server to receive the application's completion response during a later HTTP session.	**SetPort** *port_number* **AddAplURL** *url_name*	An application URL to which to send the request, and a port number at which to receive the completion of the request. For example, using the above values, issue these FNscript commands: fnscript> SetPort 5010 fnscript> AddAplURL http:sera.abbysys.com:555

Note There are no default settings for proxy HTTP client and server operations. You always explicitly configure a client proxy with an application URL and a server proxy with a port number.

For more details on the **SetPort** and **AddAplURL** commands, see Appendix A, "Fusion Script Commands."

Configuring Proxies to Use Java Message Service

Proxies can communicate with their partner applications by means of the Java Message Service (JMS), which is inherently an asynchronous messaging protocol. Fusion provides an interface to JMS provider applications that manage administered objects (such as topics and queues) through the Java Naming and Directory Interface (JNDI). Fusion access to the administered objects is also through JNDI.

Before you can configure a proxy to use JMS, you must install and configure the following third party components:

Java Runtime Environment The required version of the Java Runtime Environment (JRE) must be installed on the node hosting the Fusion proxies. Refer to the Platform Matrix, available at http://www.forte.com/support/platforms.html for the version of JRE supported for this release. Additionally, the JAVA_HOME environment variable must point to the JRE installation, as indicated in the JRE documentation.

JMS Provider Application You must install a JMS provider application supported for this release of Fusion. Refer to the Platform Matrix, available at http://www.forte.com/support/platforms.html, for a list of the JMS provider applications supported for this release. The JMS provider application must be configured to make administered objects available through JNDI. It is recommended that you run and test the JMS provider application, with the required administration, before configuring Fusion proxies to use JMS.

JMS Provider Environment You must provide Fusion with enough information to properly set the classpath to access the client libraries of your JMS and JNDI providers. The default classpath supports access to the Java Message Queue (JMQ) using the file system or LDAP JNDI providers. (However, you must properly set the JMQ_HOME environment variable for each node that accesses JMQ.) You can use the following Fusion Script commands to set the classpath to other JMS or JNDI providers:

```
AddToJMSClassPath specified_path
RemoveFromJMSClassPath specified_path
```

Table 2 shows the FNscript commands used to configure a Fusion backbone and proxies for JMS. For additional information on using these commands, refer to Appendix A, "Fusion Script Commands" on page 73.

Table 2 *Configuring Proxies to use JMS*

Proxy Role	FNscript Command	Description

All Proxies	**SetProtocol** Name=JMS
	JNDIInitCtxtFactory=*JNDIInitCtxtFactory*
	JNDIURL=*JNDIURL*
	[JNDIUser=*JNDIUser*
	JNDIPassword=*password*]
	QueueCF=*QueueCF* | TopicCF=*TopicCF*
		Configures JNDI settings for access to the JMS provider application.
		This command is available at the backbone level, to configure all proxies in a backbone, and also at the proxy level to configure individual proxies.
		Parameters:
		▮ *JNDIInitCtxtFactory*
		JNDI initial context factory. Mnemonics are available for specifying access to file, LDAP, COSNS, and RMI context factories. Refer to the Fusion Script documentation for this command for more information.
		▮ *JNDIURL*
		JNDI provider URL
		▮ *JNDIUser / JNDIPassword*
		Only required for simple authentication
		▮ *QueueCF*
		JNDI lookup name for queue connection factory
		▮ *TopicCF*
		JNDI lookup name for topic connection factory

Sender	**AddAplURL** jms:*JNDIName*:MessagingDomain=(PointToPoint | PublishSubscribe) **SetAplSession** user=*JMSUsername* password=*JMSpassword*	**AddAplURL** configures a proxy to the named JNDI provider, specifying the messaging type (either point-to-point or publish/subscribe). This command is available at the proxy level only. The argument to this command is a continuous string with no white space. The following optional parameters can be specified for **AddAplURL**. Each optional parameter is separated by the slash '/' delimiter character. ▮ JMSType=*anyString* Defaults to "Fusion" ▮ DeliveryMode=(Persistent | NonPersistent) Default value is Persistent ▮ TimeToLive=Msec Override default values ▮ *type*Property(*Name*=*Value*) *type* can be Boolean, Byte, Double, Float, Int, Long, Short, or String *Name* is any valid JMS property name *Value* is a string representation of the specified type. Note: To specify JMS authentication for a proxy, use the **SetAplSession** command.

Listener	**AddListener** Protocol=JMS
	Domain=(PointToPoint | Subscribe)
	Destination=*JNDILookupName*
	MessageSelector=*JMSmsgSelExp*
	AckMode=(auto | client)
	RemoveListener Protocol=JMS
	Destination=*IncomingJMSDestName*
		Add or remove listeners from a proxy. These commands are available at the proxy level only. This release supports only one listener per protocol.

Process Engine Session User Profile

Process-based proxies transmit user profile information to be validated by the Fusion process engine before a session can be opened with the engine. This information includes user profile name, user name, password, and roles. The values are stored in an encoded form in the proxy's configuration file; the proxy provides them to the engine at runtime to establish a session. These properties are not configured for independent proxies.

Application Session User Profile

HTTP session authentication allows the proxy to authenticate clients or to provide authentication to applications that request it. With this feature enabled, only authorized users are able to obtain access. For details on how session authentication is provided, see "HTTP Session Authentication" on page 33.

An HTTP client proxy is configured differently for session authentication from an HTTP server proxy. For details, see "Configuring HTTP Proxies" on page 54.

Configuring a Backbone

When you create a new backbone, a template for the backbone is created and saved. You then enter a series of commands to complete the configuration.

The configuration file is managed persistently on the Forte central server node by the FNscript utility. You use FNscript to create new configurations, update existing configurations, and display configurations.

Note Backbone manager and proxy nodes and their names must be added to the Forte environment following installation.

Adding a New Backbone

The following procedures show how to set up a sample Fusion environment with the following characteristics and properties:

- A process-based backbone called *Orders*, whose proxies handle different activities in an order processing workflow.
- Fusion engine properties, using an engine called *EngineA*.
- A proxy called *StartAndVerify*, which is a client/server. Its application both creates and verifies new orders.
- Two Fusion roles defined in the Fusion user profile as *OrderStart* and *OrderVerify*.
- Two proxy stylesheets with the filenames *NewOrderIn* and *VerifyOut*.

If you want to write a script to simplify the configuration process, see Appendix B, "General Script Commands."

Configuring Basic Properties

Note FNscript is not case-sensitive.

To configure a new backbone:

1. Start Fusion Script at a command prompt:

```
fnscript >
```

2. Create a new Fusion backbone by simply giving it a name.

```
CreateBackbone Orders
```

3. Find the backbone (make it current):

```
FindBackbone Orders
```

4. Specify whether the backbone interacts with a Fusion engine (for process-based integration, the default) or not (independent proxies).

   This sample backbone is process-based, so the following step is optional:

```
UseProcessEngine on
```

5. Name the Fusion engine for this backbone:

```
SetCEngine EngineA
```

   The specified Fusion engine does not need to be running until you start the backbone. Step 5 is not needed if UseProcessEngine is set to **off**.

6 Find the backbone manager (to make backbone manager commands available):

```
FindManager
```

7 Specify the backbone manager node:

```
SetNode Pluto
```

The configuration file now contains the backbone name, Fusion engine properties, backbone manager node, and a file version number.

Registering XSL Stylesheets

You can register Fusion XSL stylesheets for a backbone. You do not need to have configured all the proxies before doing so. When you register an XSL stylesheet, a persistent copy of the stylesheet is saved in the same directory as the configuration file for the backbone. This directory is on the Forte central server. FNscript and its supporting services locate and maintain the directories that contain Fusion configuration information.

Note Registration using FNscript is only needed if you are not registering stylesheets using the XML/XSL Plan Workshop, as described in the Fusion Backbone online help.

To register a stylesheet for the backbone using FNscript:

1 Find the backbone if it is not current:

```
fnscript> FindBackbone Orders
```

2 Register a stylesheet by providing the path and filename and optional stylesheet name. If you omit the stylesheet name, the stylesheet name becomes the same as the filename without a suffix. In the example, two stylesheets are set:

```
RegisterRule c:\stylesheets\NewOrderIn.txt NewOrder
RegisterRule c:\stylesheets\VerifyOut.txt Verify
```

The **RegisterRule** command has an [**overwrite**] option to use when replacing an old stylesheet with a new stylesheet by the same name. The [**updateproxy**] option is required if you register a stylesheet while any proxy in the backbone is running. This option updates all running proxies with information on the stylesheet change.

3 Display the stylesheets you have registered:

```
ListRules
```

For more information on managing stylesheets, see "Managing XSL Stylesheets" on page 59. For the procedure on proxy configuration, see "Setting XSL Stylesheets for a Proxy" on page 51.

Registering SSL Certificates

SSL is an option available for HTTP proxies. For proxies that use SSL, you can register individual certificates or sets of certificates at the backbone level and later specify the SSL options for the proxy. You do not need to have configured all the proxies before doing so. When you register a certificate, a persistent copy is saved in the same directory as the configuration file for the backbone. This directory is on the Forte central server.

To register a certificate for the backbone using FNscript:

1. Find the backbone if it is not current:

   ```
   fnscript> FindBackbone Orders
   ```

2. Register a certificate by providing the path and filename and optional certificate name. If you omit the certificate name, the certificate name becomes the same as the filename without a suffix. In the example, the file could contain either an individual certificate or a set of certificates that defines the acceptable roots:

   ```
   RegisterCertificate c:\certificates\OrdersJRS.txt
   ```

   The **RegisterCertificate** command has an [**overwrite**] option to use when replacing an old certificate with a new certificate by the same name. The [**updateproxy**] option is required if you register a certificate while any proxy in the backbone is running. This option updates all running proxies with information on the certificate change.

3. Display the certificates you have registered:

   ```
   ListCertificates
   ```

   For more information on managing certificates, see "Managing SSL Certificates" on page 60. For the procedure on proxy configuration, see "Specifying Proxy SSL Usage" on page 56.

Adding and Configuring Proxies

Within the backbone, you add proxies by naming them. When you add a named proxy, empty slots are created in the configuration file to contain the subsequently configured properties. You can display configuration details at any time using **ShowConfiguration**.

You add a new proxy from the backbone level. Configuration details differ somewhat for proxies that do not use a process engine, as noted in the following procedure.

To add a proxy to a backbone:

1. Find the backbone if it is not current:

   ```
   FindBackbone Orders
   ```

2. Add a proxy to the backbone and name it appropriately:

   ```
   AddProxy StartAndVerify
   ```

3. Make the proxy current:

   ```
   FindProxy StartAndVerify
   ```

4. Specify the proxy node:

   ```
   SetNode Pluto
   ```

5. If the proxy is independent (it does not use a Fusion process engine), turn off the default usage of the process engine. (This step is not necessary if the backbone engine property is also off.)

   ```
   UseProcessEngine off
   ```

To change autostart and autorestart properties:

1. If you do not want the proxy to be started automatically when the backbone starts, set the AutoStart property to **off**. (This property is **on** by default. No change is needed if this proxy should start automatically.)

```
SetAutoStart off
```

2. If you do not want the proxy to be *restarted* automatically after it fails, set the AutoRestart property to **off**. (This property is **on** by default. No change is needed if the backbone manager should automatically restart this proxy.)

```
SetAutoRestart off
```

To specify Fusion engine and session usage:

The following three steps specify Fusion engine and session usage, which is only applicable to process-based proxies. If the proxy is not process-based, skip to “Setting XSL Stylesheets for a Proxy.”

1. Specify the Fusion roles for validating the Fusion user session. Each role must also be defined in your Fusion user profile. The role specifies the operations the proxy is allowed to perform. In the following example, two roles are set:

```
AddCRole OrderStart
AddCRole CreditVerify
```

Note Fusion does not validate that you registered the Fusion role information with the Fusion engine.

2. Specify the remaining Fusion engine session information. The proxy uses this information to open a session with the Fusion engine and thereby control the activities offered to the application. The user name is required and the password is optional:

```
SetCSession JoeAnn
```

3. Specify the name of the Fusion user profile:

```
SetCProfile OrdersUP
```

Setting XSL Stylesheets for a Proxy

After stylesheets are registered with the backbone you apply specific stylesheets to a proxy by naming the stylesheet and setting the direction in which it applies. For details on backbone registration, see “Registering XSL Stylesheets” on page 49.

You specify one of the following options for a stylesheet when you apply it to a proxy:

- [**in**] = inbound only (documents from application to proxy)
- [**out**] = outbound only (documents from proxy to application)
- [**both**] = both inbound and outbound

Each proxy can have a maximum of two stylesheets: one inbound and one outbound. (If you specify the [**both**] option, the proxy uses only one stylesheet.) XSL stylesheets are optional, but they are almost always included in a proxy configuration.

To set a registered stylesheet for a specific proxy:

1. Find the proxy:

```
FindProxy OrderEntry
```

2. Set the stylesheet and its property. In the example, both an inbound and an outbound stylesheet are set:

```
SetRule IN NewOrder
SetRule OUT Verify
```

3. To display the configuration details:

```
ShowConfiguration
```

Setting Recovery and Reliability Options

Fusion provides various commands to configure efficient and reliable communication between proxies and their partner applications. Using these commands you can:

- achieve a level of application load balancing, by having a proxy access the application at multiple locations
- specify values for retrying communication with an application
- ensure that an application is always available to its proxy
- achieve load management for a proxy by setting limits on concurrent sessions and timeout values

To balance loads for an application:

1. Provide the proxy with additional URLs for the application, using a separate command for each URL you want to add. (Find the proxy first, if it is not current.)

```
AddAplURL http://www.Xena.mycocom:688
```

2. Set the value of the **SetAplUsage** command to **roundrobin**.

```
SetAplUsage roundrobin
```

When the proxy needs to communicate with the application, it uses each URL in the list in turn. In this way, the load on the application is shared across each application instance for which you have specified an application URL.

To ensure application availability:

1. Use the **AddAplURL** command to provide the proxy with a list of URLs for the application, using a separate command for each URL you want to add. (Find the proxy first, if it is not current.)

```
AddAplURL http://www.titan.olympiacom:688
```

2. Set the value of the **SetAplUsage** command as desired:
   - Use **failover** to use the additional URLs only if the first location is inaccessible.
   - Use **roundrobin** to use the additional URLs for load balancing, also, as described above.

```
SetAplUsage failover
```

If an application has recovered from failure, you can resynchronize its status with its proxy so that it resumes work on the activities that were active before failure. For information on resynchronizing an application and proxy, see “Resynchronizing Applications and Proxies” on page 61.

To specify application retry procedures:

1. Use the **AddAplURL** command to specify each URL the proxy should use to access the application. (Find the proxy first, if it is not current.)

   ```
   AddAplURL http://www.Jupiter.mycocom:688
   ```

2. Use the **SetAplRetry** command and include the following arguments:
   - **count** to specify how many times to try to access the application
   - **wait_time** to specify how many seconds to wait between each attempt
   - **wait_time_change** to increase the time of each wait period
   - **failure_action** *action* to specify what the proxy should do if it cannot access the application (**abort** the current activity, **rollback** to the previous activity, or **shutdown** the proxy).

   ```
   SetAplRetry Count=3 Wait=60 WaitChange=15 OnFail=rollback
   ```

   The proxy attempts to contact the application at each URL according to the wait values you have specified; it does not execute the failure action until it has tried all URLs for the application the number of times specified.

To specify session limit values:

1. Use the **SetSessionMaximum** command to specify the maximum number of concurrent sessions for a proxy.

   Specify whether this limit applies to inbound (to the proxy) sessions, outbound (to the application) sessions, or both. (Find the proxy first, if it is not current.)

   ```
   SetSessionMaximum both 200
   ```

2. Use the **SetSessionTimeout** command to specify how many seconds an application session should be held while other applications are waiting for sessions. The timeout feature is only applicable to a proxy that is acting as an HTTP server or client/server.

   ```
   SetSessionTimeout 120
   ```

   If you are configuring a JMS proxy, see the next section for additional steps. For details on HTTP proxy configuration, see “Configuring HTTP Proxies” on page 54.

Configuring JMS Proxies

This section shows one way to configure a proxy to use JMS. (You also need to specify the configuration details as described in the preceding section.)

To configure all proxies in a backbone to use JMS:

1. Find the backbone you want to configure. This example uses the Orders backbone, created previously.

   ```
   FindBackbone Orders
   ```

2. Specify the JNDI settings for the JMS provider application, using the simple authentication scheme. This example configures listener proxies using point-to-point messaging. For more information on JNDI authentication and JMS messaging models, refer to “JMS Sessions” on page 35.

   ```
   SetProtocol Name=JMS
               JNDIInitCtxtFactory=com.sun.jndi.ldap.LdapCtxFactory
               JNDIURL=ldap://mydomain.com:389/o=OrderCenter
               JNDIUser=username JNDIPassword=pwd
               QueueCF=OrderCenterQCF
   ```

3 Find a proxy and configure it as a listener that uses NewOrder as a selector property. For more information on JMS message properties and selectors, refer to "JMS Message Selectors" on page 36.

```
FindProxy OrderProcess
AddListener Protocol=JMS MessagingDomain=PointToPoint
            Destination=OrderCenterQ
            "Selector=Activity LIKE 'NewOrder'"
            AckMode=client
```

4 Find a proxy, and configure it as sender that specifies NewOrder as a selector property.

This parameter is a continuous string without white space

```
FindProxy OrderEntry
AddAplURL
 jms://cn=OrderCenterQ:MessagingDomain=PointToPoint
/StringProperty(Activity=NewOrder)
```

Configuring HTTP Proxies

An HTTP proxy must be configured as an HTTP server, HTTP client, or both. (You also need to specify the configuration details as described in the section, "Adding and Configuring Proxies" on page 50.)

▶ **To configure a proxy as an HTTP server or client:**

1 If the proxy is an HTTP server, specify the proxy HTTP port number on which it should listen for requests:

```
SetPort 6500
```

2 If the proxy is an HTTP client (its partner application is a service provider), specify the application URL for this proxy:

```
AddAplURL http://www.Jupiter.mycocom:688
```

HTTP proxies can optionally be configured to support HTTP session authentication and enable SSL (Secure Sockets Layer) on the connection, as explained in the following sections.

Specifying HTTP Session Authentication

You can configure a proxy to automatically provide or request session authentication information. For an overview of session authentication, see "HTTP Session Authentication" on page 33. Different FNscript commands are used to configure an HTTP client proxy and an HTTP server proxy; certain commands are also reserved for independent proxies that make use of session authentication.

Client proxy session authentication

A proxy acting as an HTTP client, which must be process-based, can provide session authentication information to a service provider that requires it. For background details, see "Application Session User Profile" on page 47. The following procedure explains how to enable a client proxy to automatically provide authentication when appropriate.

Note An application might require an authentication document that only specifies a user profile name and a role, even though other information can also be included. For a description of the authentication document vocabulary, see the Fusion Backbone online help.

To configure an HTTP client proxy for session authentication:

1 Use the **SetAplSession** command to specify user name, password, and other information for authenticating a proxy-to-application request. (Find the proxy first, if it is not current.)

```
SetAplSession John xyz manager
```

Only the user name is required. The [other_info] argument can contain any information that the application requires for validation, for example, the user’s department.

2 Use the **AddAplRole** command to specify a role for the user requesting the session:

```
AddAplRole ApproveCredit
```

The role can be any role that is meaningful to the application.

3 Use the **SetAplProfile** command to specify a user profile name for validating the proxy’s request for the HTTP session:

```
SetAplProfile Manager
```

Server proxy session authentication

A proxy acting as an HTTP server can be process-based or independent. It can require session authentication information from a service requestor application. The following procedures show how to enable a server proxy to request either Basic or FusionXML authentication. Basic authentication is the only option for independent proxies; process-based proxies can use either authentication scheme.

To specify the Basic authentication scheme:

1 Specify Basic as the scheme type:

```
SetAuthentication Scheme=Basic
```

The Basic scheme tells the proxy to require a *name:password* combination from the application.

2 By default, the proxy uses the process engine for authentication. If the proxy is independent, you must specify that credentials be verified on the local machine:

```
SetAuthentication Server=Local
```

3 Specify the credentials to be used for a local authentication:

```
SetCredentials User=Xavier98 Password=798xxYU
```

To specify the FusionXML authentication scheme:

The other scheme, FusionXML, tells the proxy to require an XML authentication document containing additional user validation information. Only process-based proxies can use this scheme.

1 Specify FusionXML as the scheme type:

```
SetAuthentication Scheme=FusionXML
```

(By default, the proxy uses the process engine for authentication; you do not have to set the server parameter to use the process engine.)

Note An application might require an authentication document that only specifies a user profile name and a role, even though other information can be included. For a description of the authentication document vocabulary, see the Fusion Backbone online help.

You can configure related session properties:

- **SetSessionTimeout** to specify how long the proxy waits for the application to respond to a request for authentication
- **SetSessionMaximum** to specify how many concurrent sessions a proxy accepts

For more details about setting session management values, see "Setting Recovery and Reliability Options" on page 52.

Specifying Proxy SSL Usage

HTTP proxies can optionally use SSL to secure the proxy-application connection. By default SSL is turned off. When you enable SSL, you can also specify the levels of secrecy, integrity, authentication, and compression, as well as whether sessions should be automatically resumed, and the usage of certificates. Defaults exist for secrecy, integrity, authentication, compression and resuming the session.

If a proxy is an HTTP server and SSL/HTTPS is required for the session with the application, the backbone must have certificates registered for it (using **RegisterCertificate**) and you must specify which certificates authenticate the proxy along with the other SSL options (using **SetSSL**). If the proxy is an HTTP client, authentication with certificates is optional. If the connection requires SSL (for secrecy and integrity but not authentication), you must specify the desired SSL options. For details on the options, see "SetSSL" on page 104.

To configure SSL options for the proxy:

1. Find the proxy:

   ```
   FindProxy OrderEntry
   ```

2. Enable SSL:

   ```
   SetSSL secured=always
   ```

3. Set the desired options. You can concatenate arguments all on one line:

   ```
   SetSSL secrecy=high integrity=high authenticate=server
   resume=always
   ```

   In the above example, the compression level is not specified because the default of None is desired.

4. If the proxy is to use certificates for authentication, specify the certificate names and the passwords that decrypt the certificate filenames:

   ```
   SetSSL cert=certXYZ cert_password=s7h389pqR roots=rootcertXY
   roots_password=Ax2y39cp8f
   ```

   The example states that the certificate to be authenticated is named certXYZ; the password to its filename is s7h389pqR; the acceptable roots is named rootcertXY; and the filename password of the file containing the list of root certificates is Ax2y39cp8f.

5. Display the configuration details (passwords will be masked):

   ```
   ShowConfiguration
   ```

Administering a Fusion Application System

Managing backbones in a Fusion application system consists of the following tasks:

- starting up and shutting down backbones
- monitoring and changing the state of backbones
- failure recovery
- managing XSL stylesheets
- managing SSL certificates, if applicable

Managing a Fusion application system involves configuration and administration tasks that require an understanding of Fusion system components and operations, and how to use the FNscript utility.

For a detailed FNscript command reference, see Appendix A, “Fusion Script Commands.”

Starting the System

Before you run your Fusion backbone, make sure that:

- the Fusion system configuration is complete
- for process-based backbones, all required process definitions and related plans are registered with the Fusion process engine, and the backbone configuration has the required engine properties

You start Fusion using Fusion Script.

To start a Fusion backbone:

1. If the backbone uses an engine, start the Fusion engine, using either the Fusion Console or Conductor Script. (For details on Fusion process engine management, see the *Forte Fusion Process Management System Guide*.)

Note For process-based backbones, you can start the Fusion engine before or after starting the backbone.

2. Start Fusion Script (FNscript) from a command prompt:

```
C:\> fnscript
```

3. Start the backbone (you can start any specified backbone without having to find it first):

```
StartBackbone Orders
```

The backbone manager starts, along with any of its proxies configured to AutoStart (the default).

Process-based proxies initialize their local XSLT processor and contact the Fusion engine specified for the backbone. These proxies use their Fusion engine session, role, and profile parameters to open the session. After the Fusion engine session is established, normal proxy operations begin, including appropriate communication with the application, such as establishing session authentication. If a Fusion process activity is offered to the proxy, it attempts to start the activity.

Shutting Down the System

You can shut down the backbone or any of its components. If you want to control how and when proxies shut down, it is best to use the proxy shutdown options (see the next section "Shutting Down a Proxy.").

To shut down a Fusion backbone:

1. Start Fusion Script from a command prompt:

   ```
   C:\> fnscript
   ```

2. Find the backbone you want to shut down:

   ```
   FindBackbone Orders
   ```

3. Shut down the backbone:

   ```
   Shutdown
   ```

   Each proxy exits after suspending its Fusion engine session. The Fusion backbone manager also exits.

Note The backbone is still current.

4. Shut down the Fusion engine if it is not being used by another backbone. Use either Conductor Script or the Fusion Console to shut it down.

Note You can use the **Startup** and **Shutdown** commands at any level of the FNscript command hierarchy. There are other Start and Shutdown commands that let you specify which components should be running: **StartManager, ShutdownManager, StartProxies**, and **ShutdownProxies**. You can use these commands as follows:

- **StartManager** to start a backbone manager without starting the proxies
- **StartProxies** to run proxies while the backbone manager is shut down
- **ShutdownManager** to shut down a backbone manager while the proxies are still up
- **ShutdownProxies** to shut down proxies while the backbone manager is still up

Shutting Down a Proxy

You can shut down an individual proxy without specifying any conditions. This causes the proxy to suspend the Fusion process engine session (if one exists) and shut down. It is recommended that you use various shutdown options to control proxy behavior before and after shutdown. (These options are not available when shutting down a backbone manager or backbone.)

You can specify that the proxy:

- Wait until it is no longer processing messages or ACTIVE activities before shutting down (**Shutdown quiescent**). Before shutdown occurs, the proxy does not start any further activities. Any incoming HTTP commands are rejected with "Service unavailable" errors. For incoming JMS messages the following scenarios can occur:
  - If you are using the *publish/subscribe* message model with a *non-durable* subscription, incoming messages are lost.
  - If you are using the *publish/subscribe* message model with a *durable* subscription, incoming messages are stored by the JMS provider, and available when the proxy restarts.
  - If you are using the *point-to-point* messaging model, incoming messages are stored by the JMS provider, and are available when the proxy restarts.

When you show the status (**ShowStatus**), the proxy state is indicated as “Shutdown when no activity (quiescent).” You can use **Shutdown quiescent** along with the [**restart**] and [**terminate**] options.

Note If any activity still in the ACTIVE state has a long life cycle, shutdown may not occur for some time after issuing a **Shutdown quiescent** command.

- Shut down after ### seconds (**Shutdown after=###**). This option causes the proxy to shut down after a specified number of seconds. For example, **Shutdown after=600** shuts the proxy down after 10 minutes. When you show the status (**ShowStatus**), the proxy state is indicated as “Shutdown pending” if the time limit has not been reached. You can use **Shutdown after=###** with the [**restart**] and [**terminate**] options.
- Restart immediately after shutdown (**Shutdown restart**), as long as the proxy is configured for autorestart (the default) and the backbone manager is up. If you use the [**quiescent**] and [**restart**] options together, the proxy waits until it is no longer processing messages or ACTIVE activities before shutting down, and after shutdown immediately restarts.
- Terminate the proxy’s session with the process engine after shutdown (**Shutdown terminate** or **Shutdown quiescent terminate**). A warning message displays when you use the [**terminate**] option alone; to proceed with a shutdown and terminate you have to reissue the command (**Shutdown terminate**). Any ACTIVE activities are aborted. A safer alternative is to use **Shutdown quiescent terminate**, which causes the proxy to shutdown and terminate the session only when there are no ACTIVE activities or messages in the proxy’s queue.
- Cancel out of a pending shutdown (**Shutdown cancel**), if there is still time, for example, if you previously issued a **Shutdown quiescent** or **Shutdown after=###** command.

Managing XSL Stylesheets

Each proxy has stylesheets associated with it. The stylesheets are contained within a stylesheet document. The stylesheet base for the backbone contains all the documents registered for the backbone. These files are stored and managed in the rules subdirectory on the central server node’s FORTE_ROOT directory. For example:

```
FORTE-ROOT\sysdata\fusion\c10\myrepos\mybackbone\rules
```

You might need to periodically change the proxy’s stylesheets, as well as the stylesheet base for the backbone.

Note For details on using the XML/XSL Workshop to create and debug stylesheets and manage stylesheet collections, see the Fusion Backbone online help. This section describes how to use FNscript to manage stylesheets.

Managing backbone stylesheet registration

The backbone must be current when you make changes to the stylesheet base.

- To list all current stylesheets in the stylesheet base, use **ListRules**.
- To display the contents of a stylesheet, use **ListRule** *stylesheet_name*.
- To add a new stylesheet to the backbone stylesheet base, use **RegisterRule**. You can update existing stylesheets while proxies are running using the [**updateproxy**] option. You can overwrite an old stylesheet with a new one using the [**overwrite**] option.
- To remove a stylesheet from the stylesheet base, use **DeRegisterRule**. Shut down all proxies before you use this command, or make sure only the backbone manager is up and use the [**updateproxy**] option to control the change.

Managing proxy stylesheets

A proxy can only have two stylesheet documents in total. A stylesheet document contains only one type of stylesheet: the stylesheet is inbound, outbound, or both.

The proxy must be current when you make changes to its stylesheet configuration.

- To add a new stylesheet document to the proxy, use **SetRule**. See "Configuration Options" on page 40.
- To remove a stylesheet document from the proxy, use **ClearRule**. This action removes the stylesheet document of the type specified by the parameter: in, out, or both.
- To add, change, or remove an individual stylesheet within a stylesheet document, you have to create a new document to replace the old document.

Managing SSL Certificates

If the backbone has HTTP proxies that use SSL, and certificates are used for authentication, the certificate files must be maintained. Each proxy has certificates associated with it. A certificate file contains either an individual certificate or a set of acceptable roots. The certificate base for the backbone contains all the certificates registered for the backbone. These files are stored and managed in the certificates subdirectory on the central server node's FORTE_ROOT directory. For example:

```
FORTE-ROOT\sysdata\fusion\c10\myrepos\mybackbone\certificates
```

You might need to periodically change the proxy's certificates, as well as the certificate base for the backbone.

Managing backbone certificate registration

The backbone must be current when you make changes to the certificate base.

- To list all current certificates in the certificate base, use **ListCertificates**.
- To add a new certificate to the certificate base, use **RegisterCertificate**. You can update existing certificates while proxies are running using the [**updateproxy**] option. You can overwrite an old stylesheet with a new one using the [**overwrite**] option.
- To remove a certificate from the certificate base, use **DeRegisterCertificate**. Shut down all proxies before you use this command, or make sure only the backbone manager is up and use the [**updateproxy**] option to control the change.

Managing proxy certificates

You specify which certificates apply to a proxy at the component level of FNscript. The proxy must be current when you make changes to its certificates.

- To add a new certificate to the proxy, use **SetSSL cert**=*certificate_name*.
- To specify the password that decrypts the certificate filename use **SetSSL cert_password**=*certificate_password*.
- To add a new set of acceptable roots, use **SetSSL roots**=*root_certificates*.
- To specify the password that decrypts the roots certificate filename use **SetSSL roots_password**=*password*.

Monitoring and Recovery

The Fusion backbone manager handles configuration updates while the system is online. It also monitors proxy status and restarts proxies automatically if they fail (unless the AutoRestart value is set to **off**).

Note You can run a proxy without having the backbone manager up. However, if you want the backbone manager to monitor proxy status, make sure it is running.

You can invoke a **ShowStatus** command from any level in the command hierarchy to display status information for that level. For example, to see whether there is a pending shutdown on a proxy, issue the command at the component level and look at the Proxy State line.

Resynchronizing Applications and Proxies

If a proxy or its partner application has failed and recovered, Fusion requires that you resume activities from the point of failure. To resynchronize the application and proxy so they both know what they were working on, they need to send the appropriate XML messages.

Note Resynchronizing state between an application and proxy is only possible when proxies are process-based and hence use the Fusion engine to maintain state.

Recovering from Application Failure

If an application participating in a Fusion business process fails and recovers, it should contact the proxy for a list of activities that were in the ACTIVE state when the failure occurred. The proxy constructs a list of currently ACTIVE activities upon request. To have the proxy construct this list, the application must send an instruction that maps to the command document ListActivities command. For details on the application recovery use case, see the *Forte Fusion Backbone Integration Guide*.

Recovering from Proxy Failure

When the Fusion backbone manager restarts a proxy after failure, a process-based proxy reconnects to the Fusion engine and checks the list of currently ACTIVE activities. For each activity, the proxy generates an ActivityExists state document.

You can have the proxy notify the application of its activities that were active, so that the application can take appropriate action. For details on the proxy recovery use case, see the *Forte Fusion Backbone Integration Guide*.

Chapter 4

Proxy Documents

This chapter explains the purpose and the types of XML documents processed or generated by the Fusion application proxy. Proxy documents specify the operation of the Fusion backbone by controlling message flow between the proxy and participating applications, as well as between the proxy and the Fusion process engine.

For a detailed reference of proxy document vocabulary, and a discussion of XSLT as supported in Fusion, see the Fusion Backbone online help. For details on developing XSL stylesheets, see the *Forte Fusion Backbone Integration Guide*.

Proxy Document Types

A proxy document is an XML document that the application proxy processes to fulfill its role in brokering and transforming messages flowing between applications.

XSL stylesheets defined for the proxy determine the types of documents that are generated or processed and the resulting actions.

The proxy processes four types of proxy documents:

Command Document

- The command document, generated by the proxy, is either a command document containing instructions for the Fusion process engine, or an application command document. A Fusion engine command document is used only by process-based proxies. An application command document contains instructions for sending a document to the application.

State Document

- The state document, generated by the proxy, conveys relevant state change information from the Fusion process engine (for example, that a specified process or activity has started). The result of a state document is a command document that sends the information to the application.

Authentication Document

- The authentication document is provided in response to a request, either by the application or the proxy, for validation of a user session. Authentication documents apply only to proxies configured for HTTP communication.

Application Document

- The application document, determined by the application integrator, is sent or received by the application. The contents of these documents may or may not be processed by the proxy, according to the XSL stylesheets in effect. When the proxy applies its XSL stylesheets to an incoming application document, it may generate a command document. Events (change of state information) originating from the engine are converted into state documents and result in command documents that determine and send an appropriate application document to the application.

The command document, state document, and authentication document have defined structures and vocabularies.

Note Appropriate stylesheets must be provided for command, state and application documents. No stylesheets are needed to process the authentication document. The mechanism for handling authentication requests and responses is provided by HTTP cookies.

Figure 7 shows the origination and routing of the types of proxy document that use XSL stylesheets:

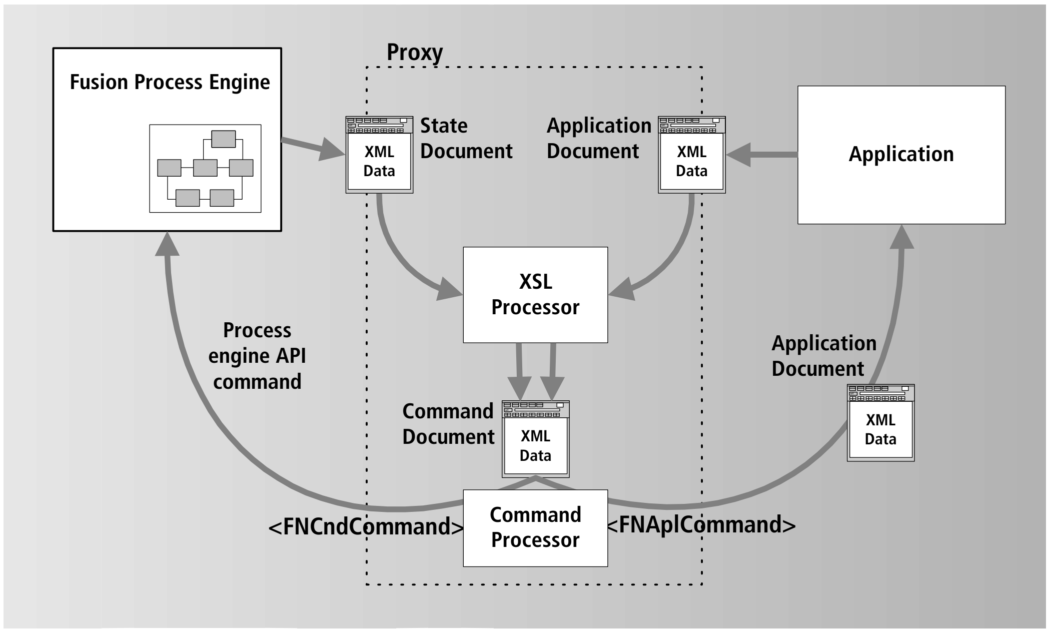

Figure 7 *Origination and Flow of Proxy Documents*

Flow of Proxy Documents

Figure 7 can be used to interpret various scenarios in the management of a business process.

Service requestor application

If the application is a service requestor, a typical flow of documents is as follows (starting at the upper right of the diagram):

- The service requestor sends an application document to the proxy to request the creation of a process instance.
- The proxy's XSLT processor transforms the application document into a command document containing the CreateProcess instruction for the Fusion process engine. The proxy invokes the process engine's CreateProcess API.
- The engine reports changes of state, which the proxy converts to a state document. If it successfully created the process instance, the proxy generates a ProcessStarted state document.
- The proxy's XSLT processor generates an application command document from the state document. The possible commands are either SendMessage or SendResponse. The command document sends an appropriate application document back to the service requestor that the application can interpret as a successful creation of the process instance.

Service provider application

If the application is a service provider, the flow of documents originates with the proxy (starting from the middle of the diagram). The following steps describe a typical synchronous interaction between the proxy and a service provider application:

- The proxy sends a command document to request that the process engine start an activity.
- The activity transitions to the READY state.
- The proxy generates an ActivityStarted state document, and a command document that will send the request to the service provider.
- The application process the request and completes the activity.
- The application responds with an application document to indicate that the activity has completed.
- The proxy transforms the application document and sends a CompleteActivity command to the engine.

Understanding Proxy Document XML

The XML vocabulary of the proxy documents consists of named elements and attributes enclosed in markup. The vocabulary and structure of proxy documents are discussed in general terms here, with a detailed reference in the Fusion Backbone online help.

Note The term attribute in the following discussion typically denotes XML attributes, not Fusion engine process attributes. However, the XML vocabulary in proxy documents can be used to embed Fusion engine process attribute values.

- Anything that is not part of the markup is text or *character data*.
- Each document has a *root element* to mark its beginning and end. The root element contains one or more *child elements*. For example, in the command document illustrated in Figure 8 on page 69, FNCommand is the root element, and it has four child elements.
- Elements may be *empty* (containing no character data or nested markup).
- Some elements take *attributes* (name-value pairs).
- In the command document vocabulary, comments and no-processing instructions are declared as elements, not markup, and rendered as intact comments in output documents. Comments contained in application documents and delimited with the standard XML <!-- and --> comment tags are deleted by the parser.
- The proxy recognizes CDATA section delimiters, which escape blocks of text containing characters that would otherwise be considered markup. CDATA sections are always used for communicating Fusion process data values. CDATA sections begin with the string "<![CDATA[" and end with the string "]]>"

Elements

The root element of a proxy document, marking its beginning and end, contains one or more child elements. Elements are enclosed in tags, either matching start and end tags, or empty element tags. In the following command document example, the root element is declared with the start-tag <FNCommand> and the end-tag </FNCommand>:

Required XML declaration
Root element; start-tag
End-tag

```
<?xml version="1.0"?>
<FNCommand>
  <FNCndCommand Command="CreateProcess">
    <FNIdentity ProcessName="CreditReport"/>
  </FNCndCommand>
</FNCommand>
```

An empty element is one that contains only unnested markup and no character data. Empty tags end with a trailing "/>". In the following example, the FNIdentity element is declared with an attribute ProcessName, taking the value CreditReport. Because attribute declarations are part of markup, and there is no character data or further structure, the empty tag applies:

Empty element tag

```
<FNIdentity ProcessName="CreditReport"/>
```

The above example is equivalent to:

```
<FNIdentity ProcessName="CreditReport"></FNIdentity>
```

Attributes

Some proxy document elements can take attributes. XML attributes are name-value pairs separated by an equals sign. Attribute values are delimited by quote marks. In the following example, ProcessName is an attribute of the FNIdentity element, and it takes the value CreditReport:

Attribute example

```
<FNIdentity ProcessName="CreditReport"/>
```

Enumerated Attribute Values

Some attribute values used in proxy documents take one of a fixed set of predefined values as specified in the command and state document vocabulary. For example, in the command document vocabulary, the attribute values for FNCndCommand all correspond to the set of Fusion process engine API commands that manages processes and activities. The following example declares an attribute of type Command, with a value of CreateProcess:

Enumerated attribute value

```
<FNCndCommand Command="CreateProcess">
```

User-Defined Attribute Values

Some proxy document elements take user-defined attribute values. In the following example, Query is a string attribute of the element FNDestination, and the attribute value is user-determined:

User-defined attribute value

```
<FNDestination Query="search?q=inventory"/>
```

Comment Elements

If the proxy parses an application document containing standard XML comments, enclosed within <!-- and --> tags, the comments are deleted by the parser. The command document vocabulary provides the element FNComments as the means to preserve comments and render them in the output (proxy-generated) document. In the command document vocabulary, no-processing instruction elements (FNNoProcessing) are another type of comment element.

Process Attribute Handling

Applications in a Fusion backbone may have no direct way to share the business object. For example, different applications may store the customer record needed by the business process in unique formats in their own relational databases. This data may need to be extracted at runtime from a large message. Fusion provides the capability to share the business object as a process attribute from which to extract required data. The XSL stylesheet must map the elements used to embed the business object values in the application document to the FNProcessAttribute element in the proxy's command document vocabulary.

If the process attributes contain XML data, the engine must be able to recognize them as such and not as plain text. Process attributes containing XML data must be defined in the Fusion process definition and application dictionary as XMLData data types. An XMLData value must be mapped to the Type attribute of FNProcessAttribute. For an overview of the command document, see "Command Document" on page 69.

Application Document

The application document is an XML document sent from a participating application to the proxy, or sent from the proxy to an application. Application documents are completely user-determined. No requirements are imposed on the vocabulary and structure of application documents, other than meeting the criteria for well-formed XML; otherwise the proxy's XML parser cannot read or render it.

Some application documents are processed by the proxy's XSL stylesheets to generate a command document. The proxy can also receive and send application documents without any processing. In this case, the proxy and engine make no use of the document contents.

Command Document

The command document instructs the proxy to command the process engine (using the engine API) or send a message to an application. Other uses of the command document are to add comments or no-processing instructions.

The child elements of the root FNCommand element are either command elements (FNCndCommand or FNAplCommand) or comment elements (FNComment or FNNoProcessing). Unlike standard XML comments enclosed with <!-- and --> tags, these comments are not deleted by the parser and are rendered in the generated proxy document.

FNCndCommand attributes are either commands to the engine (for example, "CreateProcess") or parameters to those commands. (These instructions are only relevant when the proxy is process-based.)

FNAplCommand attributes allow the proxy to send messages to an application or respond to application request messages.

Figure 8 shows the structure and vocabulary of the command document:

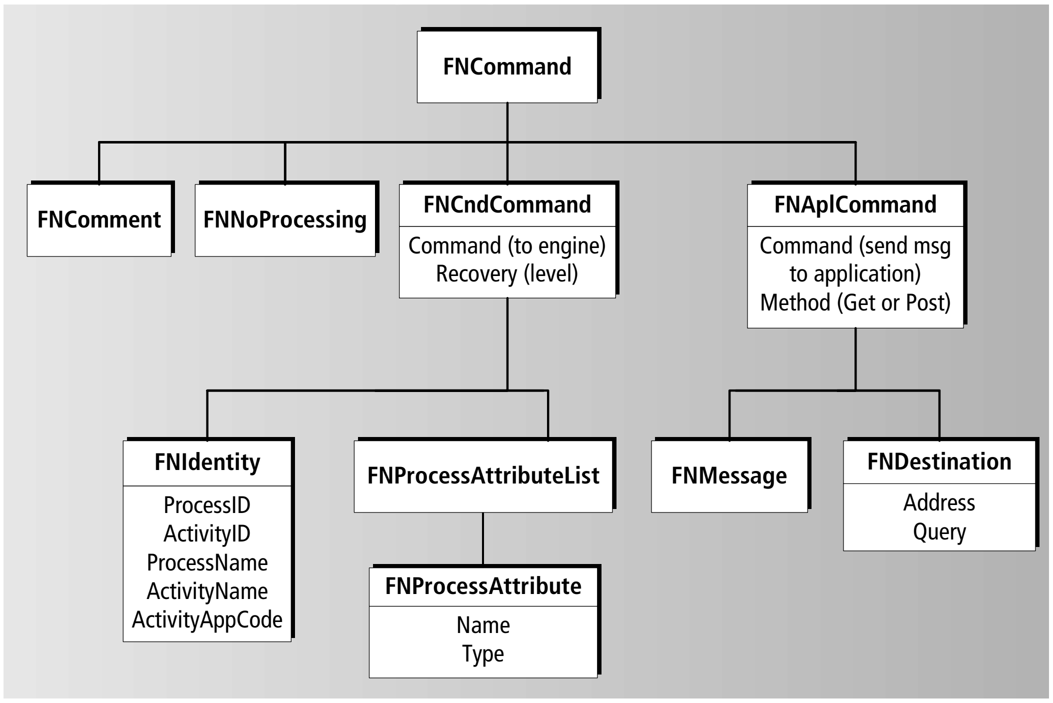

Figure 8 *Command Document Hierarchy*

The following table briefly describes each command document element:

Element Name	Description
FNCommand	root element
FNComment	comment
FNNoProcessing	code that is not to be processed
FNAplCommand	command to send message to application
FNDestination	optional element specifying destination address for the application document
FNMessage	contains an application document; child of FNAplCommand
FNCndCommand	command to Conductor engine
FNIdentity	identifies a process or activity; child of FNCndCommand
FNProcessAttributeList	process attribute list element; child of FNCndCommand
FNProcessAttribute	process attribute element; child of FNProcessAttributeList

FNCommand

FNCommand is the root element for command documents. The proxy requires FNCommand as the root element to interpret the document. FNCommand must have at least one of the following FNCommand elements as children: FNComment, FNNoProcessing, FNAplCommand, or FNCndCommand.

For a detailed reference of command document vocabulary, see the Fusion Backbone online Help.

State Document

The state document enables the proxy to send activity state change information to the application.

The state document has the following structure:

- FNState is the root element and contains the child element FNCndState.
- The child elements of FNCndState are FNIdentity and FNProcessAttributeList.

Figure 9 shows the vocabulary and structure of the state document.

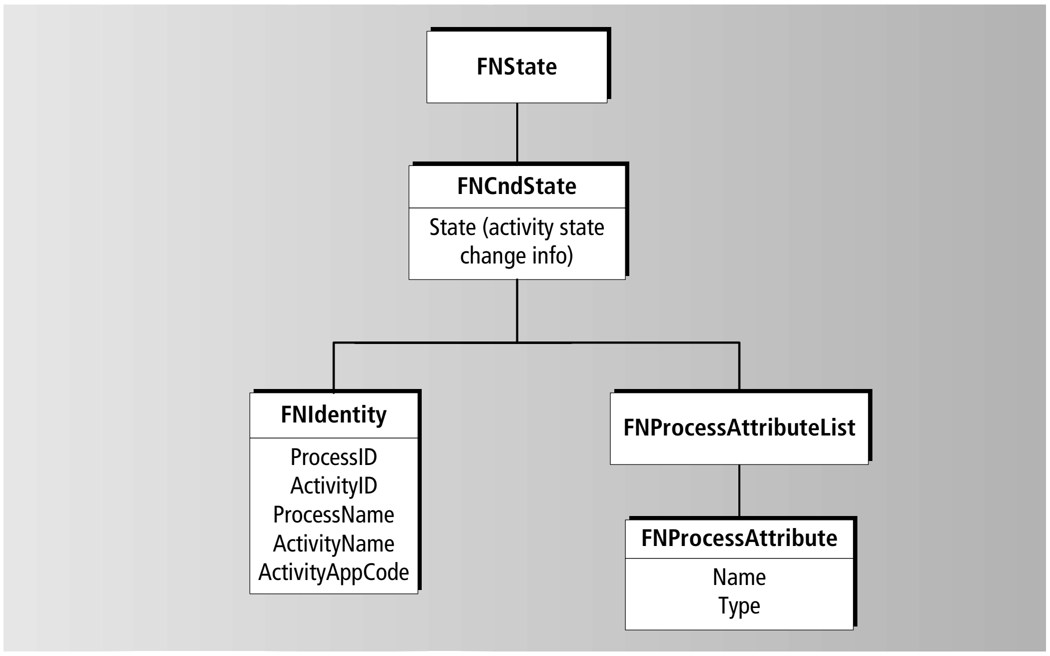

Figure 9 *State Document Hierarchy*

The following table describes each state document element:

Element Name	Description
FNState	root element
FNCndState	Conductor state change
FNIdentity	control information for a process or activity; child of FNState
FNProcessAttributeList	process attribute list element; child of FNState
FNProcessAttribute	process attribute element; child of **FNProcessAttributeList**

FNState

FNState is the root element for state documents. The proxy requires FNState as the root element to interpret the document. FNState must have FNCndState as its child element.

For a detailed reference of state document vocabulary, see the Fusion Backbone online Help.

Authentication Document

Authentication documents apply only to proxies configured to communicate using HTTP. An authentication document is provided in response to a request for validation of a user session. It can be requested either by the application or the proxy. For background on the session authentication feature, see "Proxy Concepts and Features" on page 27.

Authentication documents (and authentication in general) are optional, and all child elements within the documents are optional. Any values you provide are placed in the appropriate fields of the user profile used by the proxy.

If you use an authentication document, you provide it in the Authorization header of an HTTP message. Session authentication is managed automatically by HTTP cookies; no stylesheet is needed to interpret the authentication document.

Figure 10 shows the structure of the authentication document:

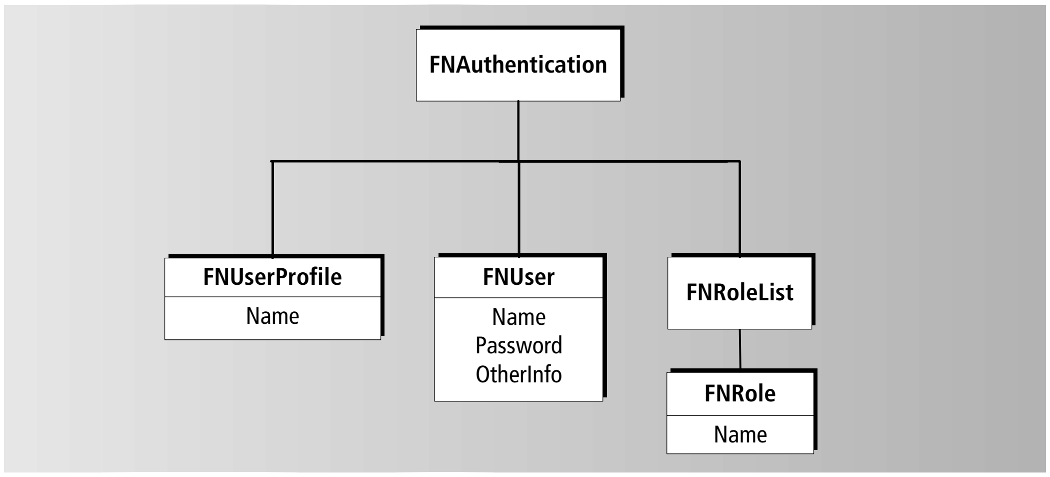

Figure 10 *Authentication Document Hierarchy*

The following table describes each authentication document element:

Element Name	Description
FNAuthentication	root element
FNUserProfile	user profile for the proxy and its partner application
FNUser	identifies a user by name, password and other optional information
FNRoleList	list of roles
FNRole	names of roles

FNAuthentication

FNAuthentication is the root element for authentication documents. The child elements are optional.

For a detailed reference of authentication document vocabulary, see the Fusion Backbone online Help.

Appendix A

Fusion Script Commands

Fusion Script (FNscript) is a utility for configuring and managing Fusion backbones and their components, namely the backbone manager and the application proxies.

This appendix explains how to start Fusion Script, how it is structured, and provides an alphabetically ordered reference of the Fusion Script commands.

Using the Fusion Script Utility

Fusion Script (FNscript) is a command-line utility that you use to configure and manage a Fusion backbone. It is the main entry point into the Fusion configuration and administration subsystem.

Fusion Script follows the conventions of Forte Conductor Script, Fscript, and Escript.

This section explains how to start FNscript and how to display online help for it.

Starting Fusion Script

You can start Fusion Script on any node in your Forte environment where the Fusion Backbone has been installed.

▶ **To start Fusion Script on any platform:**

1 Go to a system command line. For example, on a Windows NT platform:

```
C:\>
```

2 Enter the **fnscript** command at the prompt:

```
C:\>  fnscript
```

Fusion Script Help

You can use the **Help** command to display help at any time while using Fusion Script.

When you use the **Help** command with no parameters, FNscript displays the list of all available commands. This list changes as you move between FNscript operating levels. For details on these levels and how to navigate them, see “Navigating the Command Hierarchy.”

When you provide the name of a specific command as the parameter to the **Help** command (for example, **Help FindBackbone**), information about that command displays. Use the “*” character as a wildcard to obtain help for all commands that fit a pattern you specify.

Fusion Script Command Summary

The following tables summarize Fusion Script commands and group them into the following categories, each of which represents a level within the hierarchy. Each Fusion Script command is defined on one of these levels. To use a command, use the Find commands to navigate the hierarchy to the level you want to be "current," and then issue the command.

Figure 11 shows the three levels at which Fusion script commands operate.

- Environment level commands manage backbones at the highest level, for example to create, find, or delete a backbone.
- Backbone level commands set the Fusion process engine for the backbone and let you start, find and shut down proxies and backbone managers, as well as add, list, and remove proxies.
- Component level commands specify configuration values for the proxy, such as user profile, Fusion role, session authentication, URLs, or port numbers. You can also navigate from the component level up to the backbone level.

Generic commands are available at most levels so you can easily start a backbone from anywhere in the hierarchy, as well as dump status and shut down.

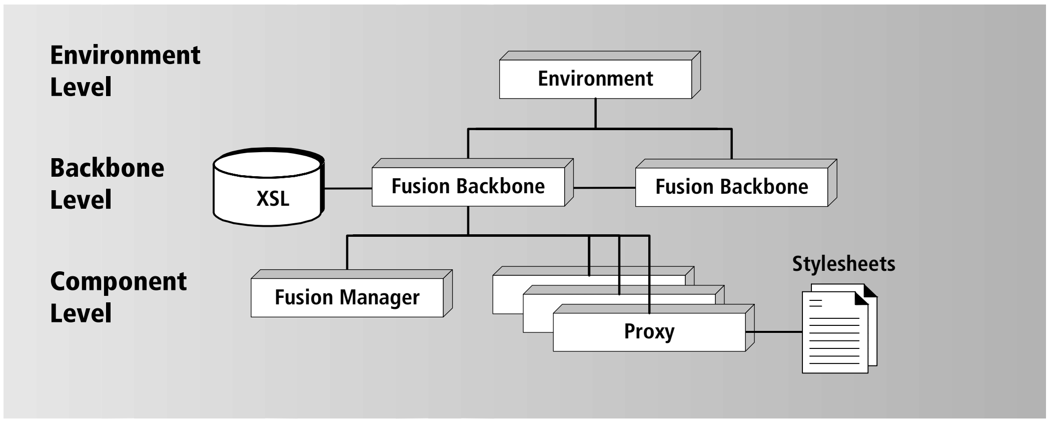

Figure 11 *Fusion Script Hierarchy*

Navigating the Command Hierarchy

There are four Find commands to navigate the FNscript command hierarchy: **FindBackbone, FindManager, FindProxy,** and **FindParent**.

- To navigate to a backbone from any level, use **FindBackbone**.
- To navigate down to a component from its backbone, use **FindManager** or **FindProxy**.
- To navigate up from a proxy to its backbone manager, use **FindParent** and then **FindManager**.
- To navigate between proxies, use **FindParent**, and then **FindProxy**.

For details on using these commands to configure and manage a Fusion application system, see Chapter 3, "Backbone System Configuration and Administration."

Generic Commands

The commands in the following table operate at most or all levels in the hierarchy. The level or component to be affected is determined by the last Find command. For example, to shut down a specific proxy from the backbone level, issue the **FindProxy** command and then the **Shutdown** command.

Table 3 *Generic FNscript Commands*

Command	Arguments	Function	See…
DumpStatus	——	Capture the lower level status information for the current proxy, backbone manager, or backbone to the log file.	page 90
ShowConfiguration	——	Display configuration detail for the current proxy, backbone manager, or backbone.	page 106
ShowStatus	——	Display the status of the current proxy, backbone manager, or backbone.	page 106
Shutdown	——	Shut down the current proxy, backbone manager, or backbone.	page 106
StartBackbone	*backbone_name*	Start the named backbone and all its components.	page 108
StartUp	——	Start the current proxy, backbone manager, or backbone	page 109
Validate	——	Validate that the configuration file contains all required entries.	page 110

Environment Level Commands

The Fusion Script environment level (base shell) commands work only in the active environment. They are used to create, delete, and list backbones and to find a specified backbone (make it current).

All the generic commands (StartUp, StartBackbone, ShowStatus, DumpStatus, ShowConfiguration, and Shutdown) also operate on the active backbone.

Table 4 *Environment Level FNscript Commands*

Command	Arguments	Function	See…
CreateBackbone	*backbone_name*	Add a backbone to the environment.	page 89
DeleteBackbone	*backbone_name*	Delete a specified backbone from the environment.	page 89
FindBackbone	*backbone_name*	Find the specified backbone. The backbone does not have to be running.	page 90
ListBackbones	*backbone_name*	Display a list of all configured Fusion backbones in the environment.	page 91

Backbone Level Commands

The Fusion Script backbone level commands apply only to the current backbone. They are available after you issue a **FindBackbone** command. Backbone level commands are used mainly for the following purposes:

- start, find, and shut down backbone managers
- add, start, find, list, shut down, and remove proxies
- register/deregister XSL stylesheets with a specific backbone, list the contents of a stylesheet, and list all registered stylesheets
- register/deregister SSL certificates with a specific backbone and list all registered certificates
- configure process engine properties for the backbone: specify an engine to use (**SetCEngine**), and allow a backbone to operate without a process engine (**UseProcessEngine)**

All the generic commands (**StartUp**, **StartBackbone**, **ShowStatus**, **DumpStatus**, **ShowConfiguration**, and **Shutdown**) are also available at the backbone level.

Table 5 *Backbone Level FNscript Commands*

Command	Arguments	Function	See...
AddListener	Protocol={HTTP | JMS} *Options* HTTP *Options* Port=*PortNumber* JMS *Options* Domain={PointToPoint | PublishSubscribe} Destination=*JNDILookupName* ["Selector=*expression*"] [AckMode=<u>auto</u> | client] [Durable=True | <u>False</u>]	Configure a proxy as either an HTTP server or JMS listener (JMS MessageConsumer).	page 84
AddProxy	*proxy_name*	Add a new proxy to the current backbone.	page 85
AddToJmsClasspath	*ClasspathEntry*	Add a jar file or directory to the JMS classpath.	page 85
ClearCEngine	——	Remove the specification for the Fusion process engine set for the backbone.	page 86
ClearListeners	——	Remove all unwanted HTTP servers and JMS listeners from the proxy.	page 87
ClearMemoryFlags	——	Clear memory settings previously specified.	page 87
DeregisterCertificate	*certificate_name* [updateproxy]	Remove the registered certificate from the backbone.	page 89
DeregisterRule	*file_name* [updateproxy]	Delete a specified stylesheet from the current backbone list.	page 90
FindManager	——	Find the current backbone manager.	page 91
FindProxy	*proxy_name*	Find the specified proxy (component level commands for the proxy are then available).	page 91
ListCertificates	*certificate_name*	List all certificates registered for the current backbone.	page 92

Table 5 *Backbone Level FNscript Commands (Continued)*

Command	Arguments	Function	See...
ListProxies	——	List all proxies configured for the current backbone.	page 92
ListRule	*rule_name*	Display contents of a named stylesheet.	page 92
ListRules	——	List all stylesheets registered for the current backbone.	page 92
RegisterCertificate	*file_name*, [*certificate_name*], [overwrite], [updateproxy]	Import a specified certificate file, containing either a single certificate or a set of trusted roots, and register with the backbone.	page 93
RegisterRule	*file_name*, [*stylesheet_name*], [overwrite], [updateproxy]	Import a specified stylesheet document and register it with the backbone.	page 93
RemoveFromJmsClasspath	*ClasspathEntry*	Remove a jar file or directory from the JMS classpath.	page 95
RemoveListener	Protocol=(HTTP Port=*PortNumber*) | (JMS Destination=*Destination*)	Remove an unwanted HTTP server or JMS listener from the proxy.	page 95
RemoveProxy	*proxy_name*	Remove a specified proxy from the current backbone.	page 96
SetCEngine	*engine_name*	Set the Fusion process engine for the backbone.	page 99
SetDocumentFormat	State=FNState | IBActionDoc	Specify the format of proxy documents to use for a backbone.	page 100
SetMemoryFlags	Flags="n:*min_pages*,x:*max_pages*"	Specify memory settings for the backbone, used as default settings for proxies in the backbone.	page 101
SetProtocol	name={HTTP | JMS} *Options* HTTP *Options* Sessions={on | off | authenticate} JMS *Options* [JNDIInitCtxtFactory=*ContextFactory*] [JNDIURL=*JNDIUrl*] [JNDIUser=*user* JNDIpassword=*password*] [QueueCF=*QueueLookupName* | TopicCF=*TopicLookupName*] [Enabled=<u>True</u> | False]	Provide configuration information for the specified protocol.	page 102
ShutdownManager	——	Shut down the current backbone manager.	page 108
ShutdownProxies	——	Shut down all proxies in the current backbone.	page 108
StartManager	——	Start the current backbone manager.	page 108
StartProxies	——	Start all proxies in the backbone.	page 109
UseProcessEngine	on | off	Instruct the backbone whether to interact with a Fusion process engine.	page 109

Component Level Commands

The Fusion Script component level commands apply mainly to the current proxy. Some commands can apply to either the backbone manager or the current proxy, for example, **FindParent, SetNode, ClearNode**.

All the generic commands (StartUp, ShowStatus, DumpStatus, ShowConfiguration, and Shutdown) also apply at this level and affect either the backbone manager or the proxy.

The following tables categorize component level commands; Table 6 lists generic commands and Table 7 lists commands used only for HTTP proxies.

Generic Component Level Commands

The commands in Table 6 summarize the usage of generic component level commands used in configuring proxies. All commands can be used, if appropriate, for an HTTP or JMS proxy. (For additional commands relating only to HTTP, see Table 7.)

Table 6 *Generic Component Level FNscript Commands*

Command	Arguments	Function	See...
AddAplURL	*URL_name*	For HTTP proxies, define one or multiple URLs for the applications that interact with a service provider proxy. For JMS proxies, configures the proxy as a JMS sender (JMS MessageProducer), supplying the necessary information to connect to a JMS provider application.	page 82
AddCRole	*role_name*	Specify a Fusion role for the current proxy.	page 83
ClearAplURL	——	Clear the specification made with AddAplURL.	page 86
ClearCProfile	——	Remove the Fusion profile for the current proxy.	page 86
ClearMemoryFlags	——	Clear memory settings previously specified.	page 87
ClearNode	*node_name*	Clear the node on which the proxy or backbone manager runs.	page 88
ClearRule	{in | out | both}	Clear a stylesheet from the current proxy.	page 88
FindParent	——	Find the parent of the current proxy or backbone manager.	page 91
RemoveAplURL	*URL_name*	Remove a specific application URL from a proxy's configuration.	page 94
RemoveCRole	*role_name*	Remove a Fusion role from the current proxy.	page 95

SetAplRetry	Count=*count* | Wait=*wait_time* |
	WaitChange=
	wait_time_change
	OnFail=*failure_action*
		Specify retry information for a proxy to use in communicating with its partner application.	page 96

SetAplUsage	roundrobin | failover	Specify how a proxy should use additional URLs.	page 97
SetAutoRestart	on | off	Set AutoRestart for the proxy on (default) or off.	page 98

Table 6 *Generic Component Level FNscript Commands (Continued)*

Command	Arguments	Function	See…
SetAutoStart	on | off	Set AutoStart for the proxy on (default) or off.	page 99
SetCProfile	*profile_name*	Specify the Fusion profile for the current proxy.	page 99
SetMemoryFlags	Flags="n:*min_pages*,x:*max_pages*"	Specify memory settings for the proxy, overriding any settings specified for the backbone.	page 101
SetNode	*node_name*	Specify or reset the node on which proxy or backbone manager runs.	page 101
SetProtocol	name={HTTP | JMS} *Options* HTTP *Options* Sessions={on | off | authenticate} JMS *Options* [JNDIInitCtxtFactory=*ContextFactory*] [JNDIURL=*JNDIUrl*] [JNDIUser=*user* JNDIpassword=*password*] [QueueCF=*QueueLookupName* | TopicCF=*TopicLookupName*] [Enabled=True | False]	Provide configuration information for the specified protocol.	page 102
SetRule	{in | out | both} *stylesheet_name*	Specify a registered stylesheet for the current proxy and indicate the direction (inbound, outbound, or both directions).	page 103
SetSessionMaximum	in | out | both *count*	Set the maximum number of allowable sessions for application-to-proxy, proxy-to-application, or in both directions.	page 103
Shutdown	quiescent | after=### restart terminate cancel	Specify an immediate shutdown or how and when the current proxy should shut down.	page 107
UseProcessEngine	on | off	Instruct the current proxy whether to interact with a Fusion process engine.	page 109

HTTP Commands

The following table summarizes the usage of commands applicable only to HTTP proxies. These commands relate mainly to the use of HTTP session authentication and SSL. To understand the concepts and implications of these commands, see Chapter 2, "Proxy Concepts and Features," and Chapter 3, "Backbone System Configuration and Administration."

Table 7 *Component Level HTTP FNscript Commands*

Command	Arguments	Function	See...
AddAplRole	*role_name*	Add a role to the information a proxy uses to authenticate itself with an application.	page 82
ClearAplProfile	——	Remove all user profile names from a proxy.	page 85
ClearAplSession	——	Remove application authentication information from a proxy.	page 86
ClearAuthScheme	——	Turn off session authentication for a proxy.	page 86
ClearCredentials	——	Remove credentials for a local authentication.	page 87
ClearCSession	——	Remove Fusion user information from the current proxy.	page 87
ClearPort	——	Clear the port number for the current proxy.	page 88
RemoveAplRole	——	Remove a role from a proxy's session authentication information.	page 94
SetAplProfile	*profile_name*	Set a user profile name for authenticating a proxy-to-application session.	page 96
SetAplSession	[user=]*name* [[password=]*password* [[otherinfo=]*other_info*]	Specify user profile information for authenticating a proxy-to-application session.	page 97
SetAuthentication	[scheme=] Basic | FusionXML [server= Local | ProcessEngine]	Specify the authentication scheme and mechanism the proxy should use.	page 98
SetCredentials	user=*name* password=*password*	Provide credentials for a local authentication.	page 100
SetCSession	[user=]*name* [[password=]*password*] [[otherinfo=]*other_info*]	Specify the Fusion user (and optional password and other information) for the current proxy.	page 100
SetPort	*port_number*	Specify the port number for the current proxy.	page 101
SetSSL	(various options; see page 104)	Specify whether to enable SSL for the proxy, and if so to what levels of secrecy, integrity, authentication and compression, and whether to resume connections. Also specify the usage of single certificates and certificate roots.	page 104
SetSessionTimeout	*timeout_value*	Set the number of seconds a proxy waits to discontinue a session if it has not received a request from an application.	page 104

Fusion Script Commands

The following is an alphabetical listing of all Fusion Script (FNscript) commands.

AddAplRole

Command Level

Component

The **AddAplRole** command specifies a role that a proxy uses to authenticate itself with an application during an outbound (proxy to application) session (for example, “CreditCheck”). The command takes effect immediately on the current proxy; you do not need to restart the proxy.

This command is only useful if the proxy is also configured (**SetAuthentication**) to use the FusionXML authentication scheme. For details on session authentication, see “HTTP Session Authentication” on page 33

Syntax

AddAplRole *role_name*

Argument	Description
role_name	The name of the role for the proxy to use.

Related Commands

To remove an application role, use the **RemoveAplRole** command.

To configure the proxy for authentication, use the **SetAuthentication** command.

Note This role has nothing to do with the Fusion role, as defined in the user validation workshop. To specify Fusion role(s) for the proxy, use the **AddCRole** command.

AddAplURL

Command Level

Component

For proxies configured for HTTP, use the **AddAplURL** command to add an application or adapter URL (Uniform Resource Locator) in the proxy configuration. The application or adapter URL is only needed if the application is a service provider. The URL in the proxy configuration must specify the location of the application. Multiple URLs may be added; the one at the top of the list is the proxy’s default partner application.

For proxies configured for JMS, use the **AddAplURL** to specify the proxy as a sender (JMS MessageProducer). When invoking **AddAplURL**, you supply a location to connect to a JMS provider, the JMS communication model, and other optional specifications.

Syntax

AddAplURL *Destination*

Argument	Description
Destination	For proxies configured for HTTP, specify the URL for the Fusion application (or adapter). For proxies configured for JMS, use the syntax described below to specify the necessary information to connect to a JMS provider application.

The **AddAplURL** command is available only when a proxy is current. Use the **FindProxy** command to make a proxy current.

For proxies configured for JMS, specify *Destination* using the following string to connect to a JMS provider application. If the specifications in the string contain white space, then place the string in quotes.

jms://*JNDIName***:MessagingDomain=**{**PointToPoint|PublishSubscribe**}*Options*

Argument	Description
JNDIName	Location of the JMS-administered object for the desired queue or topic in your JNDI namespace, using syntax appropriate for your JNDI provider.
MessagingDomain	Specifies the JMS communication model, which can be either **PointToPoint** or **PublishSubscribe**.
Options	The following series of additional specifications can be appended to the string. Each optional specification must be preceded by the slash '/' delimiter. **JMSType**=*anystring* If not specified, the default value is **Fusion**. **MessageType=**(**text**|**bytes**) Specifies whether the message should be sent as a JMS TextMessage or BytesMessage. The default is **text**. **Priority**=*1.9* Specifies the priority for the message with the JMS provider. Default priority is 4. **DeliveryMode**=(**Persistent**|**NonPersistent**) Specifies persistence for the message with the JMS provider. Default is **Persistent**. **TimeToLive**=*msec* Specifies, in milliseconds, how long the message is available to JMS listeners. 0 indicates the message does not expire. Default value is 0. *Type***Property**(*Name*=*Value*) Specifies user properties, which can be used for JMS message selection. *Type*: **Boolean**, **Byte**, **Double**, **Float**, **Int**, **Long**, **Short**, or **String**. *Name*: any valid JMS property name. *Value*: a string that can be converted to the specified type.

Related Commands

To specify how a proxy should use the URL list for an application, use the **SetAplUsage** command.

To specify JMS authentication information for a proxy, use the **SetAplSession** command.

To remove an application URL, use the **RemoveAplURL** command.

AddCRole

Command Level

Component

The **AddCRole** command adds a Fusion role to the current proxy; for example "Shipping" or "Inventory." This information, along with the user profile information, is needed for the proxy to establish a session with the Fusion process engine. The **AddCRole** command takes effect the next time the proxy is started. You can use the **AddCRole** command multiple times to add more roles to the proxy. An error message is generated if the role is already defined for the proxy.For details on Fusion roles, see the *Forte Fusion Process Development Guide*.

Syntax

AddCRole *role_name*

Argument	Description
role_name	The name of the Fusion role.

Related Commands

To remove a Fusion role, use the **RemoveCRole** command.

To specify a role to use in proxy to application session authentication, use the **AddAplRole** command.

AddListener

Command Level

Component

The **AddListener** command configures a proxy as either an HTTP server or JMS listener. Only one listener per protocol is supported.

If you are configuring an HTTP server, you specify the port number on which the proxy acts as an HTTP server.

If you are configuring a JMS listener (JMS MessageConsumer), you specify the communication model, the JNDI lookup name, and any optional arguments for connecting to a JMS provider application.

Syntax

AddListener Protocol={**HTTP | JMS**} *ProtocolArguments*

Argument	Description
Protocol=(**HTTP** | **JMS**)	Specifies the messaging protocol for the listener.
ProtocolArguments	For HTTP, specify the following for *ProtocolArguments*: **Port**=*PortNumber* Specifies the port number on which the current proxy acts as an HTTP server.
	For JMS, specify the following for *ProtocolArguments*: **Domain**={**PointToPoint** | **PublishSubscribe**} **Destination** = *JNDILookupName* ["**Selector**=*JMS message selector expression*"] [**AckMode** = **auto** | **client**] [**Durable**=**True** | **False**] **Domain**={**PointToPoint** | **PublishSubscribe**} Specifies the JMS communication model. **Destination** = *JNDILookupName* Location of the JMS-administered object for the desired queue or topic in your JNDI namespace, using syntax appropriate for your JNDI provider. For JMS, you can specify the following optional arguments for *ProtocolArguments*: ["**Selector**=*JMS message selector expression*"] Specifies which JMS properties to use for JMS message selection. [**AckMode** = **auto** | **client**] Specifies the JMS acknowledgement mode. **auto** specifies that the message is implicitly acknowledged upon receipt by the proxy; **client** specifies that the message is explicitly acknowledged only after all processing by the proxy is complete. [**Durable**=**True** | **False**] Specifies the type of registration with the JMS provider. Default specification is **False**, indicating non-durable.

The **AddListener** command is only available for the current proxy. Use the **FindProxy** command to make a proxy current.

Related Commands

You can also use the **SetPort** command to configure (or reconfigure) an HTTP server. **SetPort** *port_number* is equivalent to **AddListener protocol=http port=***port_number*.

To remove unwanted listeners from a proxy, use the **RemoveListener** command. To remove all listeners from a proxy, use the **ClearListeners** command.

AddProxy

Command Level

Backbone

The **AddProxy** command adds an unconfigured proxy to the current backbone configuration. An error is generated if the proxy has already been added to the current backbone.

Syntax

AddProxy *proxy_name*

Argument	Description
proxy_name	The name of the new proxy.

After you add a new proxy, you configure its properties. For details, see "Adding and Configuring Proxies" on page 50.

Related Commands

To verify the proxies defined for the backbone, use the **ListProxies** command.

To remove a proxy from the current backbone, use the **RemoveProxy** command.

AddToJmsClasspath

Command Level

Backbone

The **AddToJmsClasspath** command modifies your Fusion environment's JMS classpath.

Syntax

AddToJmsClasspath *ClasspathEntry*

Use *ClasspathEntry* to specify a jar file or directory (using a portable filename) to be added to the JMS classpath. You can specify environment variables (%{*varname*})in *ClasspathEntry*, which are expanded in the proxy, not FNscript.

The default configuration in your Fusion environment supports access to the Java Message Queue (JMQ) using the file system or LDAP JNDI providers. (However, you must properly set the JMQ_HOME environment variable for each node that accesses JMQ.) Use **AddToJmsClasspath** command, together with **RemoveFromJmsClasspath** to set the classpath to other JMS or JNDI providers, or otherwise modify the JMS classpath.

You can specify **jmq** for *ClasspathEntry* to add all default JMQ libraries and **sonicmq** to add all default SonicMQ libraries. This is useful for adding an extra JNDI provider to the default classpath.

Related Commands

To remove elements from the JMS classpath, use the **RemoveFromJmsClasspath** command.

ClearAplProfile

Command Level

Component

The **ClearAplProfile** command removes the definition of the profile a proxy configured for HTTP uses in an authentication document (one of the XML documents used by the proxy). The command takes effect immediately on the current proxy; you do not need to restart the proxy. For details on authentication documents, see the Fusion Backbone online help.

Syntax

ClearAplProfile

Related Commands

To specify a user profile for session authentication, use the **SetAplProfile** command.

ClearAplSession

Command Level

Component

The **ClearAplSession** command deletes the information that a proxy uses to authenticate itself with an application during an outbound (proxy to application) session. The command takes effect immediately on the current proxy; you do not need to restart the proxy.

Syntax

ClearAplSession

Related Commands

To specify authentication arguments for an application session, use the **SetAplSession** command.

ClearAplURL

Command Level

Component

The **ClearAplURL** command removes the specification previously set for the current proxy with **AddAplURL**.

ClearAuthScheme

Command Level

Component

The **ClearAuthScheme** command turns off user session authentication for a proxy configured for HTTP (thus allowing all user requests to be accepted). The setting takes effect immediately on the current proxy; you do not need to restart the proxy. this command is ignored by proxies configured for JMS.

Syntax

ClearAuthScheme

Related Commands

To turn on session authentication and specify the authentication mechanism and scheme, use the **SetAuthentication** command.

ClearCEngine

Command Level

Backbone

The **ClearCEngine** command clears any specification for the process engine to be used for the current backbone. The command takes effect the next time the backbone starts.

An engine is required to manage any business processes defined in the process definition for the backbone. You can use **SetCEngine** to specify an engine for the backbone.

Syntax

ClearCEngine

Note **SetCEngine**, automatically clears the previous engine specification when it specifies a new engine. You do not have to first issue **ClearCEngine**.

ClearCProfile

Command Level

Component

The **ClearCProfile** command removes a Fusion user profile name from the current proxy. For details on Fusion user profiles, see the *Forte Fusion Process Development Guide*.

Syntax

ClearCProfile

Related Commands

To specify a Fusion user profile name, use the **SetCProfile** command.

To clear the user name, password, and other information, use the **ClearCSession** command.

ClearCredentials

Command Level

Component

The **ClearCredentials** command removes the credentials set for a service provider proxy that uses local authentication, as specified using **SetAuthentication** and **SetCredentials**.

Syntax

ClearCredentials

Related Commands

To specify credentials for local authentication, use the **SetCredentials** command.

To specify the authentication scheme and mechanism, use the **SetAuthentication** command.

ClearCSession

Command Level

Component

The **ClearCSession** command removes Fusion user information for the current proxy.

Syntax

ClearCSession

Related Commands

To specify user information for the current proxy, use the **SetCSession** command.

ClearListeners

Command Level

Component

The **ClearListeners** command removes all unwanted HTTP servers or JMS listeners from the current proxy by deleting configuration information in the proxy for each server and listener. This command reverts the proxy to client-only mode (incoming messages can only come from the process engine).

Syntax

ClearListeners

Related Commands

To add listeners to a proxy, use the **AddListener** command. To remove specific listeners from a proxy, use the **RemoveListener** command.

ClearMemoryFlags

Command Level

Backbone, Component

When issued at the backbone level, **ClearMemoryFlags** command clears the memory settings previously specified for the proxy partitions in this backbone manager. When specified at the component (proxy) level, the settings for that proxy are cleared.

After calling **ClearMemoryFlags**, the default settings for the operating system are used.

Syntax

ClearMemoryFlags

Related Commands

Use **SetMemoryFlags** to specify memory settings for a backbone or proxy.

ClearNode

Command Level

Component

The **ClearNode** command clears the current node specification for the current backbone manager or proxy. It takes effect the next time the backbone manager or proxy starts.

Syntax

ClearNode

To clear the node for the backbone manager, first navigate to the backbone level and issue the **FindManager** command. To clear the node for the proxy, first issue the **FindProxy** command on a specified proxy.

You do not need to use **SetNode** before using **ClearNode**.

Related Commands

To reset the node, use the **SetNode** command.

ClearPort

Command Level

Component

The **ClearPort** command clears the port number specification, which specifies the port on which the current proxy acts as an HTTP server. This command is relevant only for service provider proxies configured for HTTP.

Syntax

ClearPort

The **ClearPort** command is only available for the current proxy. Use the **FindProxy** command to make a proxy current.

Related Commands

To set the port, use the **SetPort** command.

ClearRule

Command Level

Component

The **ClearRule** command clears an XSL stylesheet that has been specified for the current proxy.

Syntax

ClearRule {**in** | **out** | **both**}

The proxy can have exactly two stylesheets: one inbound (application to proxy) and one outbound (proxy to application). Specifying the direction the stylesheet applies establishes which stylesheet to clear.

- The [**in**] option clears the stylesheet used only for inbound documents from the application to the proxy.
- The [**out**] option clears the stylesheet used only to outbound documents from the proxy to the application.
- The [**both**] option clears the stylesheet that was set for both inbound and outbound documents.

Related Commands

To set a stylesheet for the proxy, use the **SetRule** command.

CreateBackbone

Command Level

Environment, Backbone

The **CreateBackbone** command adds a new backbone. An error is generated if the backbone already exists. The backbone configuration must be completed with other required information before it can operate. For details, see Chapter 3, "Backbone System Configuration and Administration."

Syntax

CreateBackbone *backbone_name*

Argument	Description
backbone_name	The name of the backbone to add.

Related Commands

To display a list of all configured backbones in the current environment, use the **ListBackbones** command.

To remove a backbone, use the **DeleteBackbone** command.

DeleteBackbone

Command Level

Environment, Backbone

The **DeleteBackbone** command deletes the specified backbone from the current environment. When you use the command all proxies must be down, but the backbone manager can be up.

Syntax

DeleteBackbone *backbone_name*

Argument	Description
backbone_name	The name of the backbone to delete.

Related Commands

To display a list of backbones before selecting the backbone to delete, use the **ListBackbones** command.

To add a new backbone, use the **CreateBackbone** command.

DeRegisterCertificate

Command Level

Backbone

The **DeRegisterCertificate** command deletes a specified certificate from the current backbone list. The **DeRegisterCertificate** command is only allowed when all proxies in the backbone are shut down or the backbone manager is up.

Syntax

DeRegisterCertificate *certificate_name* [**updateproxy**]

Argument	Description
certificate_name	The name of the certificate to be deregistered.

The [**updateproxy**] option is required if you deregister a certificate while any proxy in the backbone is running. This option updates all running proxies with information on the certificate change.

Related Commands

To display a list of all certificates registered with the current backbone, use the **ListCertificates** command.

To register a new certificate for the backbone, use the **RegisterCertificate** command.

DeRegisterRule

Command Level

Backbone

The **DeRegisterRule** command deletes a specified stylesheet from the current backbone list. The **DeRegisterRule** command is only allowed when all proxies in the backbone are shut down or the backbone manager is up.

Syntax

DeRegisterRule *stylesheet_name* [**updateproxy**]

Argument	Description
stylesheet_name	The name of the stylesheet to be deregistered.

The [**updateproxy**] option is required if you deregister a stylesheet while any proxy in the backbone is running. This option updates all running proxies with information on the stylesheet change.

Related Commands

To display a list of all stylesheets registered with the current backbone, use the **ListRules** command.

To display the contents of a registered stylesheet, use the **ListRule** command.

To register a new stylesheet for the backbone, use the **RegisterRule** command.

DumpStatus

Command Level

All

The **DumpStatus** command captures the detailed status for the current proxy, backbone manager, backbone, or environment to the screen or a log file. The data captured includes state information, such as pending messages, and other information useful for troubleshooting.

Syntax

DumpStatus

Related Commands

To display higher-level status information, use the **ShowStatus** command.

FindBackbone

Command Level

Environment, Backbone

The **FindBackbone** command finds the specified backbone.

Syntax

FindBackbone *backbone_name*

Argument	Description
backbone_name	The name of the backbone to be made current.

The **FindBackbone** command is normally used to check the status of the backbone (using the **ShowStatus** command) or to shut it down (using the **Shutdown** command). The **FindBackbone** command is available at the environment and backbone levels of the command hierarchy.

Related Commands

To display a list of all backbones configured for the current environment, use the **ListBackbones** command.

To start a named backbone without having to find it, use the **StartBackbone** command.

FindManager

Command Level

Backbone

The **FindManager** command finds the backbone manager for the current backbone so that you can start or shut down the backbone manager, or show the manager's status. The backbone manager need not be running.

Syntax

FindManager

Related Commands

To find a backbone, use the **FindBackbone** command.

To start a named backbone without having to find it, use the **StartBackbone** command.

FindParent

Command Level

Component

The **FindParent** command finds the proxy or backbone manager's parent.

Syntax

FindParent

The **FindParent** command lets you navigate to the backbone level from the component level, or from one proxy up through the backbone manager to another proxy. (You cannot find a proxy directly from another proxy.)

FindProxy

Command Level

Component

The **FindProxy** command finds the specified proxy. The specified proxy need not be running. The **FindProxy** command is normally the first command issued to perform any modifications to proxy properties.

Syntax

FindProxy *proxy_name*

Argument	Description
proxy_name	The name of the proxy to be made current.

Related Commands

To make a different proxy current, use **FindParent** and then **FindProxy**.

To display the list of proxies configured for the current backbone, use the **ListProxies** command.

ListBackbones

Command Level

Environment, Backbone

The **ListBackbones** command displays a list of all configured backbones in the current environment.

Syntax

ListBackbones

Related Commands

To find a specific backbone, use the **FindBackbone** command.

ListCertificates

Command Level

Backbone

The **ListCertificates** command displays a list of the certificates registered for the current backbone.

Syntax

ListCertificates

Related Commands

To register additional certificates for a backbone, use the **RegisterCertificate** command.

To delete any registered certificate from the current backbone, use **DeRegisterCertificate**.

ListProxies

Command Level

Backbone

The **ListProxies** command displays a list of all proxies which have been added to (but not necessarily configured for) the current backbone.

Syntax

ListProxies

Related Commands

To find a specific proxy, use the **FindProxy** command.

ListRule

Command Level

Backbone

The **ListRule** command displays the contents of a specified stylesheet on the screen. The stylesheet must be registered with the backbone (**RegisterRule**).

Syntax

ListRule *rule_name*

Argument	Description
rule_name	The name of the stylesheet to display.

Related Commands

To display a list of the stylesheets registered for the current backbone, use the **ListRules** command.

Note You can view a copy of a registered stylesheet using the XML/XSL Workshops. For details on working with stylesheets in these workshops, see the Fusion Backbone help.

ListRules

Command Level

Backbone

The **ListRules** command displays a list of the stylesheets registered for the current backbone.

Syntax

ListRules

Related Commands

To register additional stylesheets for a backbone, use the **RegisterRule** command.

To delete any registered stylesheet from the current backbone, use **DeRegisterRule**.

To display the contents of a stylesheet on the screen, use the **ListRule** command.

RegisterCertificate

Command Level

Backbone

The **RegisterCertificate** command imports the specified certificate file from the Fusion certificate base into the list for the current backbone. A persistent copy of the certificate is saved in the same directory as the configuration file for the backbone. The certificate file can contain a single certificate or a set of acceptable roots to be used for accepting authentication information. The registered certificates can be shared between proxies in the same backbone, but not between one backbone and another.

Syntax

RegisterCertificate *file_name [certificate_name*] [**overwrite**] [**updateproxy**]

Argument	Description
file_name	The filename for the certificate or set of acceptable roots. The filename is relative to the current working directory. Specify the full path name if your certificates will be stored in another directory.
certificate_name	The optional registration name for the certificate or set of acceptable roots within the backbone. If omitted, the filename (without the extension or path) is used as the certificate name.

The **RegisterCertificate** command is available only when a backbone is current. Use the **FindBackbone** command to make a backbone current.

The [**overwrite**] option is used when replacing an old certificate with a new certificate by the same name. It is not required if you are simply adding a new certificate.

The [**updateproxy**] option is required if you register a certificate while any proxy in the backbone is running. This option updates all running proxies with information on the certificate change.

Note There is no error message if the first few characters are the same as a certificate that is already registered.

Related Commands

To remove a certificate from the backbone, use the **DeRegisterCertificate** command.

To display the contents of a registered certificate, use the **ListCertificate** command.

RegisterRule

Command Level

Backbone

The **RegisterRule** command imports the specified stylesheet file from the Fusion stylesheet base into the list for the current backbone. A persistent copy of the stylesheet is saved in the same directory as the configuration file for the backbone. The registered stylesheets can be shared between proxies in the same backbone, but not between one backbone and another.

Syntax

RegisterRule *file_name* [*stylesheet_name*] [**overwrite**] [**updateproxy**]

Argument	Description
file_name	The filename for the stylesheet. The filename is relative to the current working directory. Specify the full path name if your stylesheets will be stored in another directory.
stylesheet_name	The optional registration name for the stylesheet within the backbone. If omitted, the filename (without the extension or path) is used as the stylesheet name.

The **RegisterRule** command is available only when a backbone is current. Use the **FindBackbone** command to make a backbone current.

The [**overwrite**] option is used when replacing an old stylesheet with a new stylesheet by the same name. It is not required if you are simply adding a new stylesheet.

The [**updateproxy**] option is required if you register a stylesheet while any proxy in the backbone is running. This option updates all running proxies with information on the stylesheet change.

Note There is no error message if the first few characters are the same as a stylesheet that is already registered.

Related Commands To display the contents of a registered stylesheet, use the **ListRule** command.

To display a list of registered stylesheets for the current backbone manager, use the **ListRules** command.

To deregister a stylesheet, use the **DeRegisterRule** command.

RemoveAplRole

Command Level Component

The **RemoveAplRole** command removes a role from the list sent to the application when session authentication is required and the authentication scheme is FusionXML. This role is then no longer available for the proxy to use to validate itself with an application. For details on session authentication schemes, see "HTTP Session Authentication" on page 33.

The command takes effect immediately on the current proxy; you do not need to restart the proxy.

Syntax **RemoveAplRole** *role_name*

Argument	Description
role_name	The name of the role to remove.

Related Commands To specify a role to use for validation, use the **AddAplRole** command.

To specify the proxy authentication method and scheme, use the **SetAuthentication** command.

RemoveAplURL

Command Level Component

The **RemoveAplURL** command removes a specific URL (Uniform Resource Locator) defined for an application (or adapter) from a proxy's configuration.

Note If there are multiple URLs defined for applications that communicate with the proxy and you remove the URL at the top of the proxy's list, the one newly placed at the top of the list becomes the proxy's default partner application.

Syntax **RemoveAplURL** *URL_name*

Argument	Description
URL_name	A URL for the Fusion application (or adapter).

This command takes effect the next time the proxy is started. The **RemoveAplURL** command is available only when a proxy is current.

Related Commands To add URLs for a proxy or adapter, use the **AddAplURL** command.

RemoveCRole

Command Level

Component

The **RemoveCRole** command removes a Fusion role from the current proxy.

Syntax

RemoveCRole *role_name*

Argument	Description
role_name	The name of the Fusion role to be removed.

The **RemoveCRole** command can be issued multiple times to remove additional roles from the proxy. For details on Fusion roles, see the *Forte Fusion Process Development Guide*.

Related Commands

To add a role to the current proxy, use the **AddCRole** command.

RemoveFromJmsClasspath

Command Level

Backbone

The **RemoveFromJmsClasspath** command modifies your Fusion environment's JMS classpath.

Syntax

RemoveFromJmsClasspath *ClasspathEntry*

Use *ClasspathEntry* to specify a jar file or directory (using a portable filename) to be removed from the JMS classpath.

The default configuration in your Fusion environment supports access to the Java Message Queue (JMQ) using the file system or LDAP JNDI providers. (However, you must properly set the JMQ_HOME environment variable for each node that accesses JMQ.) Use **RemoveFromJmsClasspath** command, together with **AddToJmsClasspath** to set the classpath to other JMS or JNDI providers, or otherwise modify the JMS classpath.

Related Commands

To add elements to the JMS classpath, use the **AddToJmsClasspath** command.

RemoveListener

Command Level

Component

The **RemoveListener** command removes an unwanted HTTP server or JMS listener from the current proxy by deleting configuration information in the proxy for the server or listener. Removing all listeners reverts the proxy to client-only mode (incoming messages can only come from the process engine).

Syntax

RemoveListener Protocol=(**HTTP Port**=*PortNumber*) | (**JMS Destination**=*Destination*)

Argument	Description
Protocol=(**HTTP Port**=*PortNumber*) | (**JMS Destination**=*Destination*)	Specifies the messaging protocol for the listener.
	For proxies configured for HTTP, use *Destination* to specify the port on which the proxy acts as an HTTP server. For example: **RemoveListener Protocol=HTTP Port=***PortNumber.* For proxies configured for JMS, use *Destination* to specify the location of the JMS-administered object for the desired queue or topic in your JNDI namespace, using syntax appropriate for your JNDI provider.

Related Commands

To add listeners to a proxy, use the **AddListener** command. To remove all listeners from a proxy, use the **ClearListeners** command.

You can also use the **ClearPort** command to reconfigure a proxy that is an HTTP server. **ClearPort** is equivalent to **RemoveListener protocol=http Port=***PortNumber*, where *PortNumber* is the port currently configured for the proxy.

RemoveProxy

Command Level

Component

The **RemoveProxy** command removes a proxy from the current backbone.

Syntax

RemoveProxy *proxy_name*

Argument	Description
proxy_name	The name of the proxy to be removed from the backbone.

The **RemoveProxy** command is available only when a backbone is current and the proxy is shut down. Use the **FindBackbone** command to make a backbone current.

Related Commands

To display a list of all proxies configured for the current backbone, use the **ListProxies** command.

To add a new proxy to the current backbone, use the **AddProxy** command.

SetAplProfile

Command Level

Component

The **SetAplProfile** command specifies a user profile to use to authenticate a proxy-to-application session for proxies configured for HTTP. The command takes effect immediately on the current proxy; you do not need to restart the proxy.

Syntax

SetAplProfile *profile_name*

Argument	Description
profile_name	The name of the user profile to be used for authentication.

Related Commands

To clear a user profile set for session authentication, use the **ClearAplProfile** command.

SetAplRetry

Command Level

Component

The **SetAplRetry** command specifies information for a proxy to use when attempting to communicate with its partner application. Values you can specify include:

- the number of times to retry the communication attempt
- how long to wait between attempts
- an additional amount of time to add to each wait
- the action to take if communication cannot be established

The setting takes effect the next time the proxy starts.

Syntax

SetAplRetry Count=*count* Wait=*wait_time* WaitChange=*wait_time_change* OnFail=*failure_action* abort | rollback | shutdown

Argument	Description
count	An integer value specifying the number of times a proxy attempts to communicate with its partner application
wait_time	The number of seconds a proxy should wait before trying again to communicate with its partner application.
wait_time_change	The number of seconds to add to the wait time each time a proxy attempts to communicate with its partner application.
failure_action	The action a proxy should take when all retry attempts to communicate with its partner application have failed. The available actions are abort | rollback | shutdown.

The **SetAplRetry** command is available only when a proxy is current. Use the **FindProxy** command to make a proxy current.

SetAplSession

Command Level

Component

The **SetAplSession** command specifies information to authenticate a proxy-to-application session for proxies. The setting takes effect immediately on the current proxy; you do not need to restart the proxy.

Syntax

SetAplSession [**user**=]*name* [[**password**=]*password* [[**otherinfo**=]*other_info*]]

Argument	Description
name	The application user name.
password	The application password for this user. If not specified, the password is assumed to be null.
other_info	Any additional information about a user, for example, the project manager name. This parameter is used only if you have specified FusionXML as the authentication method (SetAuthentication).

This command can be used for HTTP and JMS authentication. When authenticating a proxy configured for JMS, *name* is the JMS user name and *password* is the JMS password (*other_info* is not used for JMS authentication).

Related Commands

To clear validation information, use the **ClearAplSession** command.

To set the authentication scheme, use the **SetAuthentication** command.

SetAplUsage

Command Level

Component

The **SetAplUsage** command specifies how to use the list of application (or adapter) URLs (Uniform Resource Locators) in the proxy configuration:

- **failover** instructs the proxy to use an alternate URL only if the first URL is not available.
- **roundrobin** instructs the proxy to send requests to each URL in turn; the alternate URLs also are used in case of application failover.

Syntax

SetAplUsage {**failover | roundrobin**}

Argument	Description
failover | roundrobin	The policy to use for sending requests to the alternate URLs specified for an application in a proxy configuration.

The **SetAplUsage** command is available only when a proxy is current. Use the **FindProxy** command to make a proxy current.

Related Commands

To add URLs for a proxy, use the **AddAplURL** command.

SetAuthentication

Command Level

Component

The **SetAuthentication** command turns on user session authentication for a service provider proxy and specifies how to authenticate a request to connect to it. This command applies only to proxies configured for HTTP—it is ignored by proxies configured for JMS.

- The **Scheme** parameter indicates whether to use Basic or FusionXML authentication. The [**Basic**] option (as defined in HTTP 1.1) passes the user name and password; the [**FusionXML**] option (a Fusion extension) lets you send an authentication document with additional user information to the proxy. For details on authentication schemes, see "HTTP Session Authentication" on page 33.

Note **Scheme** is the default parameter; **SetAuthentication Basic** is therefore equivalent to **SetAuthentication Scheme=Basic**.

- The **Server** parameter indicates whether to use the process engine or the local computer to perform authentication. If the proxy has been set to operate without a process engine (by setting **UseProcessEngine off**) and authentication is required, the [**Local**] option must be used; the credentials must be specified using **SetCredentials**. The alternative is [**ProcessEngine**], which causes the process engine to perform the authentication on behalf of the proxy (this is the default).

Syntax

SetAuthentication [**scheme**=]**Basic | FusionXML [server= Local | ProcessEngine**]

Argument	Description
[scheme=] **Basic** | **FusionXML**	The type of session authentication to use.
[server= **Local | ProcessEngine**]	The machine to perform authentication.

The **SetAuthentication** command is available only when a proxy is current. The setting takes effect immediately on the current proxy; you do not need to restart the proxy.

Note This command replaces but is backward-compatible with **SetAuthScheme** for setting the authentication scheme (if you configured a proxy to **SetAuthScheme Basic**, that is recognized as equivalent to **SetAuthentication** [**scheme**=] **Basic**.

Related Commands

To turn off session authentication, use the **ClearAuthScheme** command.

To specify the credentials to use for local authentication, use the **SetCredentials** command.

To configure a proxy to operate without a process engine, use the **UseProcessEngine** command.

To modify session authentication settings, use the **SetProtocol** command.

SetAutoRestart

Command Level

Component

The **SetAutoRestart** command specifies whether the current proxy is automatically restarted after a failure. A proxy set to AutoRestart **on** should normally also be set to AutoStart **on**. This command takes effect immediately.

Syntax

SetAutoRestart on | off

This setting is defaulted to AutoRestart **on**. If the proxy fails, or is shut down a proxy using the [restart] option, the backbone manager restarts the proxy automatically. A proxy set to AutoRestart **off** must be made current (**FindProxy**) and started explicitly (**StartUp**) in the event of a proxy failure.

Related Commands

To specify that a proxy should autostart, use the **SetAutoStart** command.

SetAutoStart

Command Level

Component

The **SetAutoStart** command specifies whether the current proxy is automatically started when the backbone is started. This command takes effect immediately.

Syntax

SetAutoStart on | off

This setting is defaulted to AutoStart **on**. The proxy starts when the **StartUp** command is issued at the backbone or environment level. A proxy set to **AutoStart off** must be made current (**FindProxy**) and started explicitly (**StartUp**).

Related Commands

A proxy set to AutoStart **on** is not automatically *restarted* when the backbone manager detects proxy failure unless it is also set to AutoRestart **on** (the default). Use the **SetAutoRestart** command to change the autorestart setting.

SetCEngine

Command Level

Backbone

The **SetCEngine** command specifies the process engine to be used for the backbone. An engine is required to manage any business processes defined in the process definition for this backbone. The command takes effect the next time the backbone starts.

Syntax

SetCEngine *engine_name*

Argument	Description
engine_name	The name of the engine to be used for the backbone.

Note The **SetCEngine** command does not verify that the specified Fusion engine exists.

SetCProfile

Command Level

Component

The **SetCProfile** command specifies the Fusion user profile name the proxy should use when it logs in. This command is necessary if the proxy is process-based, in which case it does not start unless the user profile is registered with the Fusion engine. For details on Fusion user profiles, see the *Forte Fusion Process Development Guide*. If the proxy is being used solely for data transformation, you must specify this usage (**UseProcessEngine off**).

Syntax

SetCProfile *profile_name*

Argument	Description
profile_name	The name of the profile to be used for the proxy.

Related Commands

To clear the user profile name, use the **ClearCProfile** command.

To set the name, password, and other information, use the **SetCSession** command.

To set a proxy to operate without an engine, use the **UseProcessEngine** command.

SetCredentials

Command Level

Component

The **SetCredentials** command specifies the user name and password for a proxy to require to authenticate an application when local authentication is used. This command is typically used to configure authentication for an independent proxy configured to use HTTP. The proxy must be configured as a service provider; for details, see Chapter 3, "Backbone System Configuration and Administration." The proxy must also be set to not operate with an engine (**UseProcessEngine off**). This command is ignored by proxies configured for JMS.

SetCredentials user=*name* **password=***password*

Argument	Description
name	The user name the application should provide to the proxy.
password	The password the application should provide to the proxy. If not specified, the password is assumed to be null.

Related Commands

To clear the credentials, use the **ClearCredentials** command.

To set the authentication scheme and usage, use the **SetAuthentication** command.

To configure a proxy to operate without a process engine, use the **UseProcessEngine** command.

SetCSession

Command Level

Component

The **SetCSession** command specifies the Fusion user information for the current proxy. The user and password are validated by the Fusion process engine. This command is used for proxies that interact with a process engine.

Syntax

SetCSession [**user=**]*name* [[**password=**]*password* [[**otherinfo=**]*other_info*]]

Argument	Description
name	The Fusion user name.
password	The Fusion password for this user. If not specified, the password is assumed to be null.
other_info	Any additional information about a user, for example, backbone manager name.

Related Commands

To clear the user profile information, use the **ClearCSession** command.

To clear the user profile name, use the **ClearCProfile** command.

SetDocumentFormat

Command Level

Backbone

The **SetDocumentFormat** command specifies the XML format for proxy documents. This command is only useful if you have a reason to change the defaults.

If a backbone was created using Release 1 of the Fusion Backbone, the XML document format defaults to **IBActionDoc**. Backbones created using Fusion Backbone Release 2 default to **FNState**. For details on proxy document XML, see the Fusion Backbone help.

Syntax

SetDocumentFormat State=FNState | IBActionDoc

Argument	Description
state	The type of proxy document to use, either FNState for Fusion 2.0 XML documents, or IBActionDoc for Fusion 1.0 XML documents.

SetMemoryFlags

Command Level

Backbone, Component

When issued at the backbone level, **SetMemoryFlags** command specifies memory settings for all the proxy partitions in the backbone. It provides a default setting for all proxies in that backbone. When issued at the component (proxy) level, **SetMemoryFlags** overrides any settings specified at the backbone level.

Syntax

SetMemoryFlags Flags="n:<*minimum number of pages*>,x:<*maximum number of pages*>"

Argument	Description
Flags	The following specifications are allowed: n The minimum number of pages managed by the memory manager x The maximum number of pages managed by the memory manager

If you do not issue this command, then the default settings for the operating system are used.

SetNode

Command Level

Component

The **SetNode** command sets the node on which the current backbone manager or proxy runs. All Fusion components must have a node. The **SetNode** command takes effect the next time the backbone manager or proxy starts.

Syntax

SetNode *node_name*

Argument	Description
node_name	The name of the node in your Forte environment that hosts the proxy or backbone manager.

To set the node for the backbone manager, first navigate to the backbone level and use the **FindManager** command. To set the node for the proxy, first issue the **FindProxy** command to make the proxy current.

Related Commands

To clear the node setting, use the **ClearNode** command.

SetPort

Command Level

Component

The **SetPort** command defines or resets the port number on which the current proxy acts as an HTTP server. This command is relevant only for service provider proxies configured for HTTP.

Syntax

SetPort *port_number*

Argument	Description
port_number	The number to assign to the proxy's port.

The **SetPort** command is only available for the current proxy. Use the **FindProxy** command to make a proxy current.

SetProtocol

Command Level

Backbone, Component

The SetProtocol commands specifies configuration settings for a communication protocol, which can be HTTP or JMS.

Syntax

SetProtocol Name={**HTTP | JMS**} [*ProtocolArguments*]

Argument	Description
Name=(**HTTP** | **JMS**)	Specifies the communication protocol for the backbone or proxy.
ProtocolArguments	For HTTP, specify the following for *ProtocolArguments:* **Sessions**={**on** | **off** | **authenticate**} ▮ **on** enables HTTP sessions for the proxy regardless of whether or not the proxy is configured for session authentication. ▮ **off** turns off the default enabling of HTTP sessions for a proxy that is configured for session authentication. ▮ **authenticate** specifies that HTTP sessions are used when authentication is configured for the proxy (the default behavior).
	For JMS, specify the following for *ProtocolArguments*: [**JNDIInitCtxtFactory**=*ContextFactory*] [**JNDIURL**=*JndiUrl*] [(**QueueCF**=*QueueLookupName* | **TopicCF**=*TopicLookupName*)] [**JNDIUser**=*username* **JNDIPassword**=*password*] [**Enabled**=(<u>**True**</u> | **False**)] **JNDIInitCtxtFactory**=*ContextFactory* Specifies the JNDI initial context factory for the naming service used on site. You can specify a custom class name or any of the following mnemonics for *ContextFactory*: **file** com.sun.jndi.fscontext.RefFSContextFactory **ldap** com.sun.jndi.ldap.LdapCtxtFactory **cosns** com.sun.jndi.cosnaming.CNCtxtFactory **rmi** com.sun.jndi.rmi.registry.RegistryContextFactory **JNDIURL**=*JndiUrl* Specifies the location for the JMS-administered objects. For example, you can specify the following for *JndiUrl*: ldap://mydomain.com:389/o=JMQ **QueueCF**=*QueueLookupName* | **TopicCF**=*TopicLookupName* Specifies the lookup name for the JMS provider's connection factory. You must specify either **QueueCF** or **TopicCF**, depending on the JMS messaging model you are using (point-to-point or publish/subscribe). **JNDIUser**=*username* **JNDIPassword**=*password* Specified when connecting to a secure JNDI provider. Only simple authentication or no authentication is currently supported. **Enabled**=(<u>**True**</u> | **False**) If **True**, the proxy attempts to start the Java helper process to connect to JMS. If **False**, all JMS settings are ignored. If you specify **SetProtocol Name=JMS**, then **Enabled** is automatically set to True. If you later disable JMS, you should set **Enabled** to **False**.

This command is available at the backbone level, component (proxy) level, or both. Any proxy level settings for an argument override the backbone level settings. In an environment where all proxies use the same JNDI and JMS providers, only backbone settings are usually needed.

If a proxy is configured to use session authentication (**SetAuthentication**), HTTP sessions are enabled by default so that authentication state information is retained for the duration of the session. Without HTTP sessions, each message would need to be authorized. The **SetProtocol** command provides options for enabling and disabling HTTP sessions for a proxy.

Related Commands

To set session authentication, use the **SetAuthenticate** command.

SetRule

Command Level

Component

The **SetRule** command specifies how a stylesheet applies to XML documents a proxy sends or receives. Stylesheets must already be registered at the backbone level (**RegisterRule**). It is not necessary to clear a stylesheet (**ClearRule**) before setting a new one.

Stylesheets are applied to XML documents in one of three ways:

- inbound (documents from application)
- outbound (documents to application)
- both directions (application-to-proxy and proxy-to-application)

Syntax

SetRule {**in** | **out** | **both**} *stylesheet_name*

Argument	Description
{**in** | **out** | **both**}	Specifies whether the stylesheet applies to inbound sessions, outbound sessions, or both.
stylesheet_name	The name of the registered stylesheet to be used in the current proxy. The stylesheet name is case-sensitive and must follow the specified option.

One of the following options is required:

- The [**in**] option applies the stylesheet only to inbound documents from the application to the proxy.
- The [**out**] option applies the stylesheet only to outbound documents from the proxy to the application.
- The [**both**] option applies the stylesheet to both inbound and outbound documents.

The proxy can have exactly two stylesheets: one inbound and one outbound.

Related Commands

To register a new stylesheet for the backbone, use the **RegisterRule** command.

To clear a stylesheet specified for the proxy, use the **ClearRule** command.

SetSessionMaximum

Command Level

Component

The **SetSessionMaximum** command specifies the maximum number of concurrent inbound, outbound, or inbound/outbound user sessions a proxy accepts or uses. The setting takes effect immediately on the current proxy; you do not need to restart the proxy.

Syntax

SetSessionMaximum {**in** | **out** | **both**} *count*

Argument	Description
in | **out** | **both**	Specifies whether the count value applies to inbound sessions, outbound sessions, or both. ▮ **in** applies the session maximum only to sessions established by the application. ▮ **out** applies the session maximum only to sessions established by the proxy. ▮ **both** applies the same session maximum in either direction.
count	An integer value specifying the maximum number of sessions.

For inbound sessions, the session maximum applies only to HTTP sessions (JMS inbound sessions are currently limited to one session per proxy). For outbound sessions, the session maximum applies *separately* to each HTTP or JMS partner location.

The **SetSessionMaximum** command is available only when a proxy is current. Use the **FindProxy** command to make a proxy current.

SetSessionTimeout

Command Level

Component

The **SetSessionTimeout** command lets you conserve session resources by specifying the number of seconds a proxy acting as an HTTP server waits to discontinue a session if it has not received a request from an application and another session is required. The setting takes effect immediately on the current proxy; you do not need to restart the proxy.

Syntax

SetSessionTimeout *timeout_value*

Argument	Description
timeout_value	The number of seconds for the proxy to wait for a request from an application.

The **SetSessionTimeout** command is available only when a server proxy is current. Use the **FindProxy** command to make a proxy current.

SetSSL

Command Level

Component

The **SetSSL** command configures optional SSL usage for an HTTP proxy. By default a proxy does not use SSL. When you enable SSL, you can also specify the levels of secrecy, integrity, authentication, and compression, as well as whether connections should be automatically resumed, and the usage of certificates.

The proxy must be configured as either an HTTP server or HTTP proxy. If a proxy is a service provider (HTTP server) and SSL/HTTPS is required for the connection with the application, the backbone must have certificates registered for it (using **RegisterCertificate**) and you must specify which certificates authenticate the proxy along with the other SSL options (using **SetSSL**). If the proxy is a service requestor (HTTP client), authentication with certificates is optional. If the connection requires SSL for secrecy and integrity, you must specify the desired SSL options.

The SetSSL command has nine arguments. The first one listed below enables SSL and the next five specify usage options. Arguments can be concatenated on a single command line. The defaults are underlined:

Syntax (connection options)

SetSSL Secured=(<u>never</u> | always)

SetSSL Secrecy=(<u>any</u> | none | low | medium | high)

SetSSL Integrity=(<u>any</u> | none | low | medium | high)

SetSSL Compression=(<u>none</u> | any | speed | high)

SetSSL Resume=(<u>always</u> | never)

SetSSL Authenticate=(<u>anonymous</u> | none | server | both)

The remaining four arguments relate to using certificates for authentication. A single certificate authenticates an individual HTTP server or client. The certificates of one or more trusted roots are used for accepting authentication information. The filenames of the certificate and roots certificates files are encrypted; you use passwords to decode them for use in authenticating the session. Passwords are masked in the configuration file and do not display if you use **ShowConfiguration**. Arguments can be concatenated on a single command line.

Syntax (certificate options)

SetSSL Cert=*certificate_name*

SetSSL Cert_password=*certificate_password*

SetSSL Roots=*root_certificate*

SetSSL Roots_password=*password*

Argument	Description
Secured	Specifies whether to enable SSL or not. The default is Never. Use the Always option if this proxy requires SSL to secure the connection with the application.
Secrecy	Specifies the level of encryption for HTTP messages. Higher levels are more secure but can impact performance. The default is Any. ▮ **any** causes the highest level of secrecy to be used (High) if the partner application also specifies Any. ▮ **none** turns off secrecy ▮ **low** provides minimal secrecy (40-bit key) ▮ **medium** provides moderate secrecy (greater than 40-bit but less than 128-bit) ▮ **high** provides the highest level (128-bit or longer key)
Integrity	Specifies the type of cryptographic hash code (message digest) to use for HTTP message integrity. Higher levels offer more protection against tampering or corruption of the message. The default is Any. ▮ **any** causes the highest level of message integrity to be used (High) if the partner application also specifies Any. ▮ **none** turns off integrity ▮ **low** provides minimal integrity (currently matches no digests) ▮ **medium** provides moderate integrity (MD5) ▮ **high** provides the highest level (SHA)
Compression	Specifies the level of data compression used before encryption. SSL 3.0 does not provide any compression algorithms but requires a level. The default is None. ▮ **none** provides no compression ▮ **any** provides no compression ▮ **speed** (not supported) ▮ **size** (not supported)
Resume	Specifies whether the proxy and its partner resume a previously started SSL session without renegotiating the security parameters. This enhances system performance. The default is Always. ▮ **never** specifies that the original session should never be resumed ▮ **always** specifies that a previously started session should always be resumed
Authenticate	Specifies the level of authentication based on certificates. The default is anonymous. ▮ **anonymous** specifies that neither proxy nor application provide certificates but do negotiate keys for secrecy and integrity ▮ **none** specifies no authentication, secrecy or integrity ▮ **server** specifies that the HTTP server supply a certificate to authenticate itself ▮ **both** specifies that both the HTTP server and client supply authentication certificates
Cert=*certificate_name*	Name of the certificate to use for authentication. The certificate must be registered with the backbone.
Cert_password= *certificate_password*	The private password that encrypts the filename of the certificate.
Roots= *root_certificates*	The root certificates that are acceptable to the proxy for authenticating an application. The root certificates must be registered with the backbone.
Roots_password= *password*	The password that encrypts the filename containing the list of acceptable root certificates for the certificate.

Related Commands

To display the proxy's SSL configuration, use the **ShowConfiguration** command. The certificate and root certificate passwords are masked.

To register a new certificate for the backbone, use the **RegisterCertificate** command.

To list the certificates registered with the backbone, use the **ListCertificates** command.

ShowConfiguration

Command Level

All

The **ShowConfiguration** command displays the contents of the configuration file on the screen, with the output sorted alphabetically. The configuration information is for the proxy, backbone manager, or backbone, depending on what is current. For example, if you have issued a **FindBackbone** command, it displays configuration detail for the entire backbone. If you use the command on a proxy or backbone manager, it only displays the proxy configuration.

Syntax

ShowConfiguration

The backbone or component does not need to be running to use this command.

Related Commands

To validate configuration entries without displaying the configuration file, use the **Validate** command at any level.

ShowStatus

Command Level

All

The **ShowStatus** command is a generic command that displays status information for the current proxy, backbone manager, backbone, or environment.

Syntax

ShowStatus

Note

The Proxy State line in the configuration file is used to indicate pending shutdowns. For details, see "Proxy Shutdown Options."

Related Commands

To set the current level for which status information is needed, use a **Find** command (for example, **FindBackbone** to show status for that backbone).

Shutdown

Command Level

All

The **Shutdown** command is a generic command to shut down the current component or backbone.

Syntax

Shutdown

Use a **Find** command to make current the component or backbone you want to shut down.

The **Shutdown** command takes effect at the specified level:

- backbone level: shuts down the current backbone (backbone manager and all proxies)
- component level: shuts down only the current proxy or backbone manager (the proxy's session with the process engine is suspended)

To specify the conditions for a proxy shutdown, see the following section on proxy shutdown options.

Related Commands

To start a named backbone, use the **StartBackbone** command.

To shut down only the backbone manager and not the proxies, use the **ShutdownManager** command.

To shut down only the proxies and not the backbone manager, use the **ShutdownProxies** command.

Proxy Shutdown Options

Certain options are available when using the **Shutdown** command for the current proxy to fine-tune its behavior before and after shutdown. These options are not available when shutting down a backbone manager or backbone.

Command Level

Component

The **Shutdown** command with no options specified immediately shuts down the current proxy and suspends the session with the process engine. When you show the status (**ShowStatus**), the configuration indicates “Shutdown pending” on the Proxy State line.

The options to this command allow you to control how and when shutdown should occur.

Syntax

Shutdown [quiescent | after=*###] [restart] [terminate]*

Shutdown [cancel]

The **Shutdown** command has the following five options when applied to a proxy:

- The [quiescent] option makes the proxy wait until it is no longer processing messages or ACTIVE-state activities before shutting down. Before shutdown occurs, the proxy does not start any further activities. Any incoming HTTP commands are rejected with “Service unavailable” errors. For incoming JMS messages the following scenarios can occur:
  - If you are using the *publish/subscribe* message model with a *non-durable* subscription, incoming messages are lost.
  - If you are using the *publish/subscribe* message model with a *durable* subscription, incoming messages are stored by the JMS provider, and available when the proxy restarts.
  - If you are using the *point-to-point* messaging model, incoming messages are stored by the JMS provider, and are available when the proxy restarts.

  When you show the status (**ShowStatus**), the proxy state is indicated as “Shutdown when no activity (quiescent).” This option can be used with [restart] and [terminate], but not with the after=### option.

  Note

  If any activity still in the ACTIVE state has a long life cycle, shutdown may not occur for some time after issuing a Shutdown quiescent command.

- The [after=###] option causes the proxy to shut down after ### seconds. For example Shutdown after=600 shuts the proxy down after 10 minutes. This option can be used with [restart] and [terminate], but not with the [quiescent] option.
- The [restart] option restarts the proxy after shutdown, as long as the proxy is configured for autorestart (the default) and the backbone manager is up. If you use the [quiescent] and [restart] options together, the proxy waits until it is no longer processing messages or ACTIVE-state activities before shutting down, and after shutdown immediately restarts.
- The [terminate] option terminates the proxy’s session with the process engine. Any ACTIVE-state activities are aborted. A warning message displays when you use this option alone; to proceed with a shutdown and terminate you have to reissue the command (Shutdown terminate). A safer alternative for terminating the session is to use the [quiescent] and [terminate] options together, which causes the proxy to shutdown and terminate the session only when there are no ACTIVE-state activities or messages in the proxy’s queue.
- The [cancel] option undoes, if there is still time, a previously-issued **Shutdown** command, such as one issued with the [quiescent] or [after=###] option. The [cancel] option cannot be used with any other command options.

Related Commands

To specify that a proxy should autorestart, use the **SetAutoStart** command.

To shut down all proxies, use the **ShutdownProxies** command.

ShutdownManager

Command Level: Backbone

The **ShutdownManager** command shuts down the current backbone manager without shutting down the proxies.

Syntax: **ShutdownManager**

Related Commands: To shut down the entire backbone (backbone manager and proxies), use the **Shutdown** command.

To shut down only the proxies and not the backbone manager, use the **ShutdownProxies** command.

To start only the backbone manager but not the proxies, use the **StartManager** command.

ShutdownProxies

Command Level: Backbone

The **ShutdownProxies** command shuts down the backbone proxies without shutting down the backbone manager.

Syntax: **ShutdownProxies**

Related Commands: To shut down the entire backbone or a current proxy, use the **Shutdown** command at the appropriate level.

To shut down only the backbone manager, use the **ShutdownManager** command.

To shut down the current proxy, use the **Shutdown** command, with or without options.

To start only the proxies, use the **StartProxies** command.

StartBackbone

Command Level: All

The **StartBackbone** command starts a Fusion backbone. If you do not specify a backbone name, the command starts the current backbone.

Syntax: **StartBackbone** [*backbone_name]*

Argument	Description
backbone_name	The name of the backbone to start.

You can use this command to start a backbone from anywhere in the command hierarchy.

Related Commands: To shut down a backbone, use the **Shutdown** command at the backbone level.

StartManager

Command Level: Backbone

The **StartManager** command starts the current backbone manager.

Syntax: **StartManager**

Related Commands: To start a named backbone, use the **StartBackbone** command.

To start the current backbone and all its components, use the **Startup** command.

To shut down the backbone manager, use the **ShutdownManager** command.

StartProxies

Command Level

Backbone

The **StartProxies** command starts all proxies in the current backbone.

Syntax

StartProxies

The **StartProxies** command does not start the backbone manager if the manager is currently shut down.

Related Commands

To start the backbone manager, use **StartManager**.

To start the current backbone and all its components, use the **Startup** command.

To start a named backbone, use the **StartBackbone** command.

To shut down all proxies, use the **ShutdownProxies** command.

StartUp

Command Level

All

The **StartUp** command is a generic command to start the current backbone or component.

Syntax

StartUp

The **Startup** command takes effect at the specified level:

- component level: only the current proxy or backbone manager
- backbone level: only the current backbone manager and all proxies set to AutoStart **on**

Related Commands

To specify which proxies should autostart, use the **SetAutoStart** command.

To explicitly start a proxy which is not set to AutoStart on, issue **FindProxy** from the backbone level and then **StartUp**.

To verify that the engine has started properly, use the **ShowStatus** command.

UseProcessEngine

Command Level

Backbone, Component

The **UseProcessEngine** command specifies whether or not a backbone or a service provider proxy can interact with a Fusion process engine. The default is to use a process engine.

This command is useful for testing purposes (for example, you may want to test a proxy's transformational behavior without setting up an engine), as well as when a particular proxy or backbone is being used only for data transformation and does not participate in a Fusion-managed business process. This command is not useful if the backbone is entirely process-based (that is, no proxies in the backbone operate without an engine); you only need to specify the engine name (SetCEngine).

Syntax

UseProcessEngine on | off

At the backbone command level, **UseProcessEngine** defines usage for the current backbone, that is, the entire backbone either uses a process engine or not. At the component command level, **UseProcessEngine** defines usage for the current proxy, that is, the current proxy either uses a process engine or not.

The settings for this command do not override any engines defined using **SetCEngine**. (For example, if an engine named George is set for the backbone, that engine is only used when process engine usage is set to **on**.) However, any value set with **UseProcessEngine** for the current proxy overrides a value set with **UseProcessEngine** at the backbone level.

Caution Changes to the engine usage state are not reflected until the proxy or backbone restarts. Be sure that all activities are completed before changing the state.

Related Commands

To configure a backbone to use a specific engine, use the "SetCEngine" command.

To configure a service provider proxy that does not use a process engine for session authentication, use the **SetCredentials** and the **SetAuthentication** commands.

Validate

Command Level

All

The **Validate** command is a generic command to validate entries in the configuration file at a given level. For example, if issued at the component level, this command only validates that the required configuration of the current backbone manager or proxy is complete. If issued at the backbone level, it validates all components.

Note There might be information that is not included in the minimum requirements but is necessary for your particular backbone to operate. For example, validation does not include application URL information, but service provider proxies require this information to operate.

Syntax

Validate

Related Commands

To display the configuration file, use the **ShowConfiguration** command.

Appendix B

General Script Commands

This appendix is a reference for general script commands you can use within a Fusion application system, including commands for general operations, writing and executing scripts, file management, and for system operations.

General Script Operations

The following table lists Fscript commands you might find useful when managing your Fusion application system. These commands let you perform general operations, such as getting online help, defining the filename format, specifying a directory search path, or quitting Fusion Script:

Define format for file names

Define the directory search path

Define a command alias

Command	Arguments	Function
Exit	——	Exit Fusion Script.
Quit	——	Exit Fusion Script.
Help	[comand_name| match_string]	List help for general Fusion Script commands and currently available FNscript commands. Use a wildcard (*) to get help on similarly named commands.
ModLogger	+(log_flags)| –(log_flags)	Modify the current Forte log message filters (log flag settings) for Fusion Script.
SetOutFile	[file_name]	Specify a file where standard output is printed.
UseLocal	——	Set Fusion Script to recognize file names specified in local operating system format.
UsePortable	——	Set Fusion Script to recognize file names specified in Forte portable name format.
ShowPath	——	Show the current directory search path (see SetPath).
AddPath	directory_name [;directory_name…]	Add the specified directories to the current search path (see SetPath).
SetPath	directory_name [;directory_name…]	Set the directory search path used by any of the commands that take a file name as an argument.
ShowAlias	[alias_name]	Display one or all defined aliases with their expansions.
AddAlias	alias_name command_string	Define an alias for a Fusion Script command and its arguments.
RemoveAlias	[alias_name]	Remove an alias definition.

Writing and Executing Scripts

This section describes commands you can use to write and execute scripts that automate Fusion Script backbone management tasks. You can capture a sequence of Fusion Script operations into a script file and then run that file at a later time.

Fscript includes many more commands than are implemented in FNscript. For full reference documentation of the commands, see the *Fscript Reference Manual*.

You can run a script either by starting Fusion Script with the **-i** flag (and supplying the script file name) or by starting Fusion Script and then issuing the **Include** command when you want to run the script.

Comments

To include comments in your scripts, start the line containing the comment with the # character, as shown in the following example:

```
# Shutdown the StartAndVerify proxy
FindProxy StartAndVerify
Shutdown
ShowStatus
Exit
```

The following table lists commands most often used for writing and executing scripts:

	Command	Arguments	Function
Execute scripts	Include	filename	Execute the commands in a specified script file.
	Step	——	Step through the commands in an include script, prompting you for each command.
Write scripts	Script	filename	Capture Fusion Script commands and write them into a specified script file.
	Repeat	repeat_count	Repeat execution of the next command the specified number of times.
	Delay	milliseconds	Delay execution of the next command for the specified number of milliseconds.
	CommentOn	——	Write script file commands and output to standard output.
Print to standard output	CommentOff	——	Stop writing script file commands and output to standard output.
	ShowExpansions	show_flag	Enable or disable the printing of alias expansions to standard output when they occur.
	SilentOn	——	Turn on printing of exceptions to standard output.
	SilentOff	——	Turn off printing of exceptions to standard output.

Operating System and File Management Commands

The following table lists commands you use to interact with the operating system on a node and perform file management operations.

	Command	Arguments	Function
Work with files	ListFile	ListFile	Lists the contents of the specified file onto standard output.
	ListFiles	directory_name	List the files in a directory.
	Directory	directory_name	List the files in a directory.
	Ls	directory_name	List the files in a directory.
	WhichFile	file_name	Find the specified file in the current directory search path.
	Mv	old_file_name new_file_name	Rename a specified file in the local file system.
	Rm	file_name	Remove a specified file in the local file system.
	CopyFile	file1_name file2_name [r]	Copy a specified file in the local file system.
	Cp	file1_name file2_name [r]	Copy a specified file in the local file system.
	Duplicate	file1_name file2_name [r]	Copy a specified file in the local file system.
	Chmod	mode file_name [r]	Change the access permissions of the specified file.
	ReadIntoFile	file_name [term_str]	Read subsequent lines and write them to the specified file until the terminating string is encountered.
Define the working directory	Pwd	——	Display the name of the current working directory.
	Cd	directory_name	Change the current working directory.
	SetDefault	directory_name	Change the current working directory.
	MkDir	directory_name	Make a new directory.
Invoke a command directly on the operating system	ExecCmd	opsys_cmd [bg_flag] [in_file] [out_file] [err_file]	Execute the specified operating system command.
	ExecLocal	opsys_cmd	Execute the specified operating system command (for path names in opsys_cmd that use local format).
	ExecPortable	opsys_cmd	Execute the specified operating system command (for path names in opsys_cmd that use portable format)
	Shell	——	Start a session in which operating system commands can be invoked.
Manage memory	CollectMem	——	Run the memory reclamation (garbage collection) utility.
	MemStats	——	Dump memory statistics to the screen.
Set and print environment variables	SetEnv	variable_name, [value]	Set an environment variable to a specified value.
	PrintEnv	variable_name, [value]	Print the current value of the specified environment variable.
Show status upon exit	ExitStatus	integer	Set a return value for the session. Defaults to 0 for normal exit; 1 for abnormal exit.

Index