ArcIMS
Data Delivery Extension (DDE)
Reference Manual

Contents

About This Manual

This document describes the ESRI ArcIMS Data Delivery Extension (DDE), based on Safe Software's network-based spatial data distribution product. DDE allows a user to perform remote translation requests on a dataset and have that data transformed and translated into a wide variety of formats and delivered to the user's desktop.

The DDE is a modular extension that is configured and installed separately from ArcIMS. It consists of a number of subsystems that communicate with ArcIMS via web page integrations.

The general steps followed when installing DDE first establish the correctness of the DDE subsystem by itself, and once this is successful, then DDE is integrated with the ArcIMS framework.

About the Contents

This document discusses the components and the architecture of DDE, and how it can be deployed and configured. The supported interfaces to each of the components are described, and the means by which third-party applications can integrate with DDE to exploit its capabilities are also covered.

Configuring for user source data is described, and the system's default translation and delivery characteristics are discussed, both as an "out-of-the-box" solution, and as a model for developing customized distribution applications employing DDE .

This manual is divided into three parts:

- Part 1: Installing DDE
- Part 2: Configuring DDE for ESRI ArcIMS
- Part 3: Customizing DDE

How to Use This Manual

The headings below indicate the usual scenarios for installing DDE.

Identify the phrase in column 1 of the table below that best describes your DDE configuration requirements and then look in column 2 to determine which chapters best suit your needs. Where more than one chapter is listed in column 2, the order of listing indicates the best order in which the chapters should be read.

First-time Installation

Installing DDE	Chapter 2 – Getting Started Chapter 3 – Installing and Running DDE
Integrating DDE with ArcIMS	Chapter 4 – Introduction to DDE for ESRI ArcIMS Chapter 5 – Configuring DDE to Read User Source Data Chapter 6 – Integrating DDE with ArcIMS Chapter 7 – Enabling an ArcIMS Map for Translations Chapter 8 – Using DDE for ArcIMS

Modify an Existing Installation

Change the data source definitions	Chapter 5 – Configuring DDE to Read User Source Data

Advanced Customization

Customize DDE theme definitions	Chapter 10 – Source Data Theme Definition
Define available extract formats	Chapter 7 – Enabling an ArcIMS Map for Translations (Step 3 of *Modifying the Map Configuration Files*)
Define available extract coordinate systems	Chapter 7 – Enabling an ArcIMS Map for Translations(Step 3 of *Modifying the Map Configuration Files*)
Define multiple source datasets (e.g. Shape *and* SDE data for the same ArcIMS map)	Chapter 10 – Source Data Theme Definition Chapter 11 – Configuring Source Data Manually Chapter 7 – Enabling an ArcIMS Map for Translations (Step 3 of *Modifying the Map Configuration Files*, adding Layer to Theme Mappings)
Customize the ArcIMS DDE toolbar button functionality	Chapter 9 – Customizing the FMEExport Function

Enable extract result notification via e-mail	Chapter 14 – Translation Servlet (*Asynchronous Result Return*)
Customize data extract translation behaviour	Chapter 17 – Translation Request Processing

Conventions Used

The following conventions are used throughout this manual.

`Monospaced type`	This font is used to offset directory names, file names, operator commands and similar text.
`Pathnames`	If a pathname is listed in a generic example (not Windows- or UNIX-specific), it is shown using forward slashes “/”.

Part 1: Installing DDE

CHAPTER 1

Introduction to DDE

As spatial data increases in importance, both government and industry need to disseminate and have access to the latest data, as cost-effectively and as quickly as possible.

Historically, spatial data has been distributed using physical media, such as tape or CD-ROM. Since spatial data tends to be voluminous, a data provider would often make the data available in a single format and in a single coordinate system.

In the past, spatial data consumers had to perform a series of extra operations before the data was available for use on their system:

- Wait for the data to be delivered, then download the information from the physical media.
- Translate the data from the provided format into their system's format.
- Reproject the data from the delivered coordinate system into the preferred local coordinate system.

The translation and reprojection steps could be performed either by writing customized software or by using a commercial data translator such as Safe Software's Feature Manipulation Engine (FME).

The DDE Solution

DDE provides the critical translation and delivery component of any data distribution solution.

With DDE, Safe Software provides a complete data distribution system that reduces the effort and cost of distributing data via the Internet or Intranet. DDE includes a robust, web-enabled translation engine that provides scalable, real-time data distribution services.

Based on proven technology, DDE provides a solution that distributes data from any of the ArcIMS source data formats, while it simultaneously satisfies the needs of a user community needing data to be delivered in different formats and in different projections.

Figure 1-1 shows the high-level context within which DDE operates, and how it fits into the data distribution process.

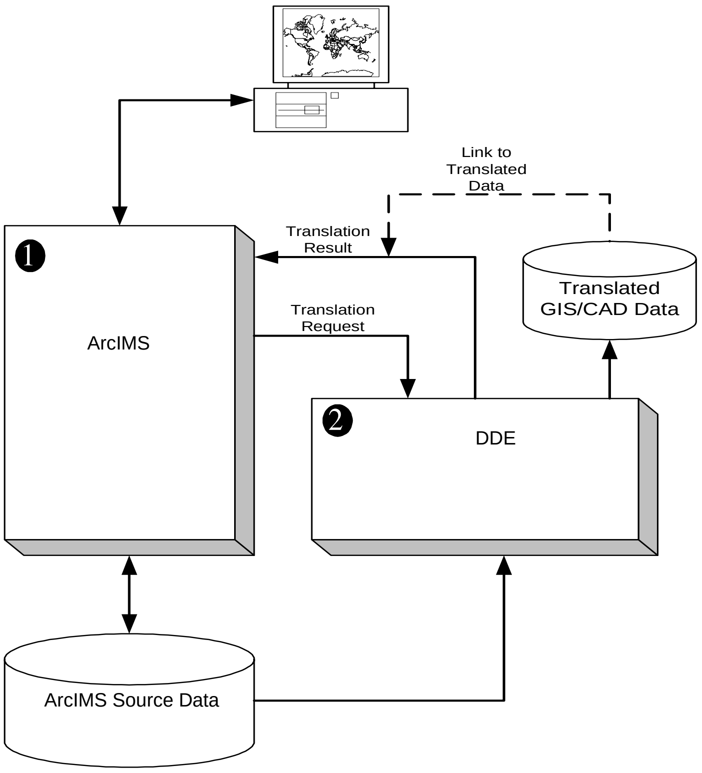


FIGURE 1-1 **High-level Context of the DDE Data Distribution Suite**

❶ *ArcIMS reads from either flat files or a live database source. The server and client components are interacting to present a view of the data source to a user interacting with a web browser.*

❷ *DDE receives requests from the client component of the web application. This request consists of a standard URL with embedded information such as the extent of interest, and the destination format and projection. The DDE takes this URL and performs the translation into the destination format. The result is then delivered to the desktop in the requested Geographic Information System/ Computer-aided Design (GIS/CAD) format.*

DDE uses web-based technologies to provide real-time distribution of spatial data, including its attributes. DDE performs translation requests by exploiting FME technology. This technology allows for the translation of spatial data to and from any of the formats and databases supported by the FME but is most effective working with industry-leading spatial databases such as ESRI SDE 3.x/ArcSDE 8.x and ArcGIS 9.0.

As shipped, DDE is configured to support the more popular GIS and CAD formats, and other formats can be easily added to custom configurations. DDE can also be set up to perform custom translations which may pull data from multiple sources. This data extraction can then be merged and translated into a single result.

An external application can request data from DDE in several ways. The simplest is through a Uniform Resource Locator (URL). The translation is performed and the results are made available to the user. Other methods involve lower-level interfacing with DDE components and are described in later chapters of this document.

All communication between DDE components takes place via the Transmission Control Protocol/Internet Protocol (TCP/IP).

Benefits of a DDE Solution

There are a number of benefits to using DDE to solve data distribution problems:

- ***Scalable:*** As the load increases on a server that is hosting DDE, more FME Translation Servers can be added on different machines over a network. This allows many processing resources to be added to the system to meet peak demands.
- ***Real-Time Translation:*** DDE performs real-time translations using Safe Software's industry-leading FME Translation Server.
- ***Customizable:*** The DDE QServer and FME Server APIs are published, flexible and robust. As a result, third-party components easily integrate, allowing for full exploitation of DDE's capabilities.
- ***Data Independence:*** DDE can be configured to distribute data from any format or system supported by its underlying FME Translation Server.
- ***Data Security:*** All data access is specified through configuration and FME mapping files. These files define which data is accessed and made available to the client applications. Users of the remote applications have no direct access to the data.

How DDE Works

DDE is a set of cooperating software components that can be configured to distribute spatial data over a network to external applications making requests for that data. This section discusses how each of the components forms a link in a chain that couples the user's desktop to the source data, and what each component is dependent upon in the environment.

Components

DDE consists of the following main components:

- FME Server (FME licensed to run in its server mode)
- QServer
- Translation Servlet
- Process Monitor

DDE is designed to work in tandem with ArcIMS, as shown in Figure 1-2. All communication between ArcIMS and DDE is either via a URL compliant with the http protocol, or via TCP/IP using the published DDE component APIs.

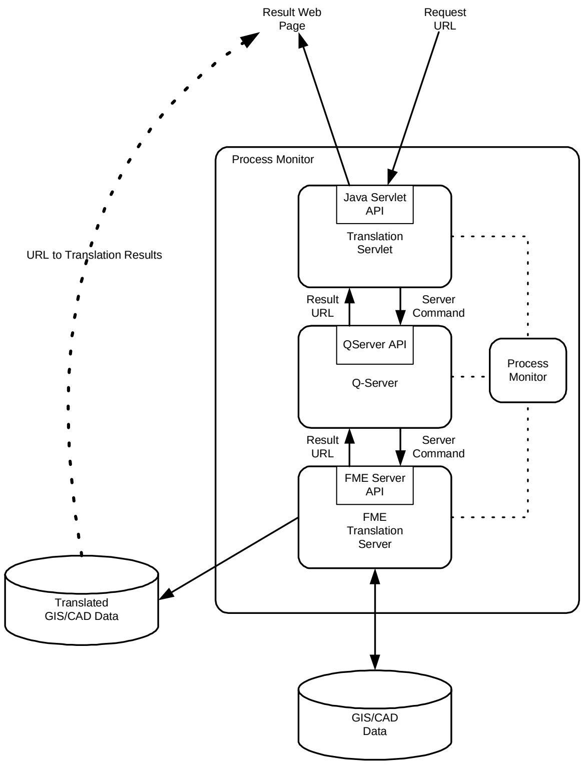


FIGURE 1-2 **DDE Components**

As a result of this approach, ArcIMS and DDE components can all reside on a single machine or they can be distributed to different machines across a network. A detailed description of DDE components is provided in subsequent sections of this manual. A high-level explanation of the DDE component architecture is shown below.

FME Translation Server

The FME Translation Server (hereafter referred to as *FME Server*) is the most sophisticated component. It is responsible for performing the translation and making the translation result available to the calling application. This component extracts and translates the data out of the underlying data store, and it exploits all of the FME's capabilities.

Figure 1-3 shows a sample configuration where each DDE component executes on a separate processing resource. This is only one example of a configuration. This figure highlights the dependencies the components have on the system software and, in addition, shows how the components can be distributed or combined. Each of the rectangles represents a processing resource and, in each case, the supporting operating system may be either Windows or UNIX.

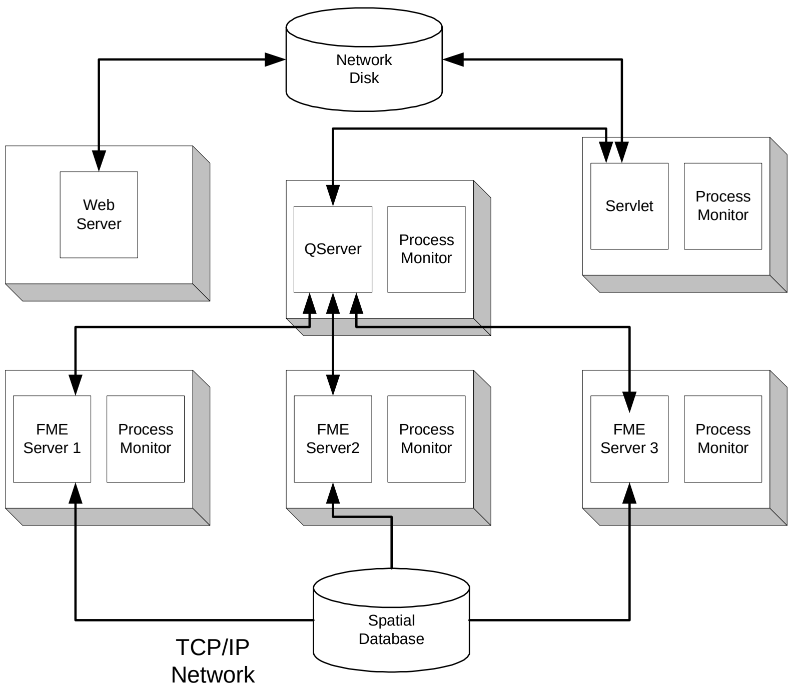


FIGURE 1-3 **DDE Configuration Example**

Unlike the standard FME version, when the FME Server is launched, it goes into server mode where it awaits translation requests. All interaction between the FME Server and the outside world takes place via the FME Server API. This API enables applications written in C, C++, and Java to send translation requests directly to the FME Server.

Since the FME Server is always running and is initialized, it is capable of performing real-time data translation.

The FME Server :

- ***Register Mode:*** The FME Server registers itself with another application such as the QServer. In this way multiple FME Servers can be used, greatly increasing the system's data translation throughput. This is the mode used by DDE.
- ***Standalone Mode (not applicable to DDE):*** The FME Server starts up and waits for clients to connect to it with translation requests. Then it processes the translation request and returns the result to the client. In this mode only one client at a time is serviced by the FME Server and that client must disconnect before another client can connect.

In the DDE environment, one (or more) FME Server is always connected to an external application that feeds it translation requests. From an architectural standpoint, it is important to note that the FME Server has no direct relationship or knowledge of this external application. The application is using the FME Server API. The FME Server receives requests from the application, processes them, and returns the results.

The FME Server has an API available in Java, C, and C++ enabling third parties to incorporate the FME Server into their own applications.

QServer

Safe Software developed the QServer as a management layer application. Its purpose is to manage all translation requests to DDE.

The QServer is a Java component that provides the scheduling layer to DDE, ensuring that no FME Server is idle while there are translation requests that have not been started. When a translation request arrives and all FME Servers are busy, the QServer queues the request, thereby ensuring that it gets serviced as quickly as possible.

The QServer provides the following functionality to DDE:

- ***Load Balancing:*** The translation requests form a load that is balanced among the available FME Server instances to ensure that translation requests are handled as soon as possible.
- ***Performance Monitoring:*** Collects statistics such as length of time needed for a translation, average turnaround time, maximum turnaround time, etc. These statistics are useful for determining when the system has reached capacity during peak times. This information is available to third-party programs via the QServer API.
- ***Dynamic Resource Addition:*** FME Server resources can be added to or removed from the system at any time. There is no need to shut down the

system when adding or removing processing resources. This is useful for meeting peak load demands of the system.

- ***Scalable Solution:*** Provides the scalable layer enabling a large number of FME Server components to be connected and pooled to provide a truly scalable data distribution system.
- ***Fault Tolerance:*** If an FME Server is killed or terminated while sending a request, the QServer automatically puts the request to the front of the translation queue and sends it to the first available FME Server. This ensures that requests are not lost during normal system operation.

As with any other application that converses with DDE, the QServer has no knowledge of other DDE components and merely follows a well-defined communication protocol. The QServer uses a stateless protocol, and each request received from the client application and forwarded to the underlying FME Server is self-contained. The QServer has its own well-defined API, which is independent from the FME Server, available in C, C++, and Java.

Translation Servlet

The Translation Servlet (hereafter referred to as the *Servlet*) is an application of Sun's Java Servlet technology. It operates at the front-end of the DDE system and provides an interface for translation requests from client applications such as web-based mapping systems.

An external application sends a translation request encoded as an http-compliant URL string to the Servlet. The URL includes parameters describing the request's various details. The Servlet builds a translation command from this URL and sends it on to the QServer for processing.

Once translation results are available, the Servlet sends back to the calling application an HTML web page that includes a URL pointing to the results. Result delivery is then accomplished as a standard web server download to the calling application via the URL reference.

Process Monitor

The Process Monitor (hereafter referred to as the *Monitor*) is a program that starts other programs and automatically restarts them if they terminate for any reason. This enables a degree of fault tolerance to be added to a system.

Environments having one or more programs that need to be continuously running can have these programs started and monitored by the Monitor. If any program terminates, aborts or crashes for any reason, the Monitor will automatically restart it without user intervention.

DDE component relationships are shown in Figure 1-4.

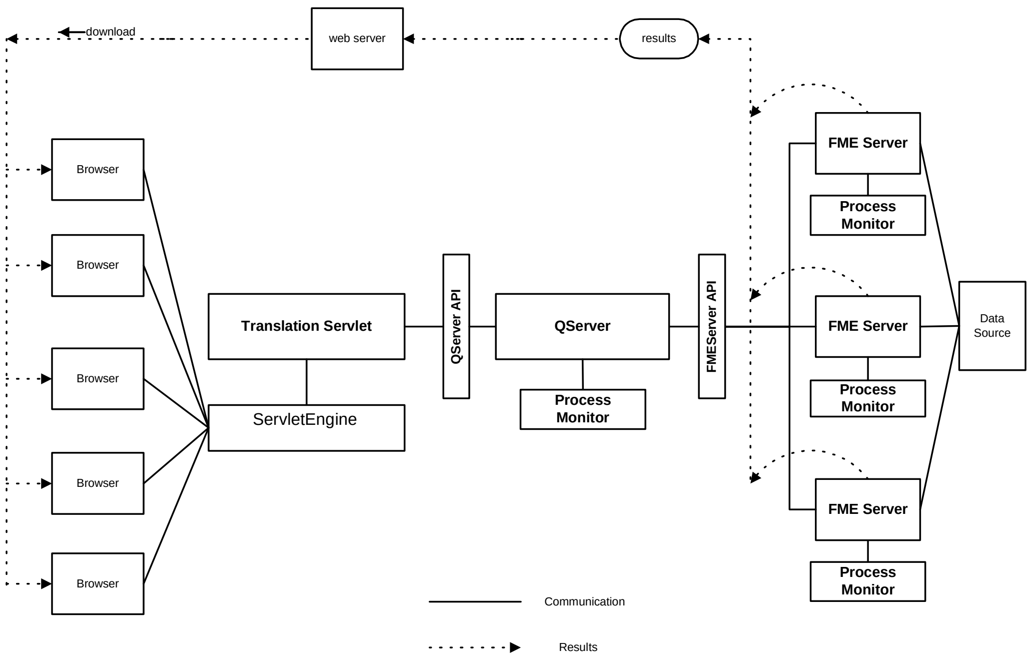


FIGURE 1-4 **Component Relationships**

General Sequence of Operations

From a user’s perspective, interfacing with DDE is done via ArcIMS and the download form communicating with the underlying DDE system.

Users interact with ArcIMS to visually select the map layers and region that they desire. They may then use the DDE download form to further customize a subset of features requested from the source dataset, and the desired format and projection.

Upon request submission, the DDE download form passes the request to the Servlet in the form of a request URL. The Servlet receives and converts the URL into an FME Server command that it passes on to the QServer. The QServer waits for the first available FME Server to which it can pass the translation request.

The FME Server receives the request, accesses the requested data from the source dataset, translates it to the required format and projection, and stores the result. The translation operation itself is controlled by the collection of FME mapping files.

Depending on the success or failure of the translation, the FME Server either sends a success or failure message back to the caller, in this case the QServer. If the translation failed, an appropriate failure message will be returned. If the translation was successful, then a request completion status message will be returned. This message consists of the status and an embedded URL that points to the translation results.

The QServer forwards this status message, without any changes, on to the Servlet. If, for example the translation was successful, the Servlet will create a results web page to deliver to the user's browser. This page is composed from the HTML templates files. If the request was for a zipped result, then the Servlet fetches the remote fetch zip template. It replaces the embedded keywords with the URL returned by the QServer. The resulting web page contains an active link. The user can then click on this link to download the actual translation result file.

When the user downloads results by clicking on the page link, the browser responds by sending a standard download request to the web server on the system specified in the link. The web server accesses the results and downloads them to the user's system.

It is important to note that it is the request's completion status that the FME Server returns to the QServer, which is then returned to the Servlet. The translation status is returned with an included URL – not the actual translation results. The delivery of the results is accomplished as a file download or GIF display between the browser and the web server via a link supplied by the Servlet in the web page.

Figure 1-5 illustrates this entire sequence of events as a timing diagram.

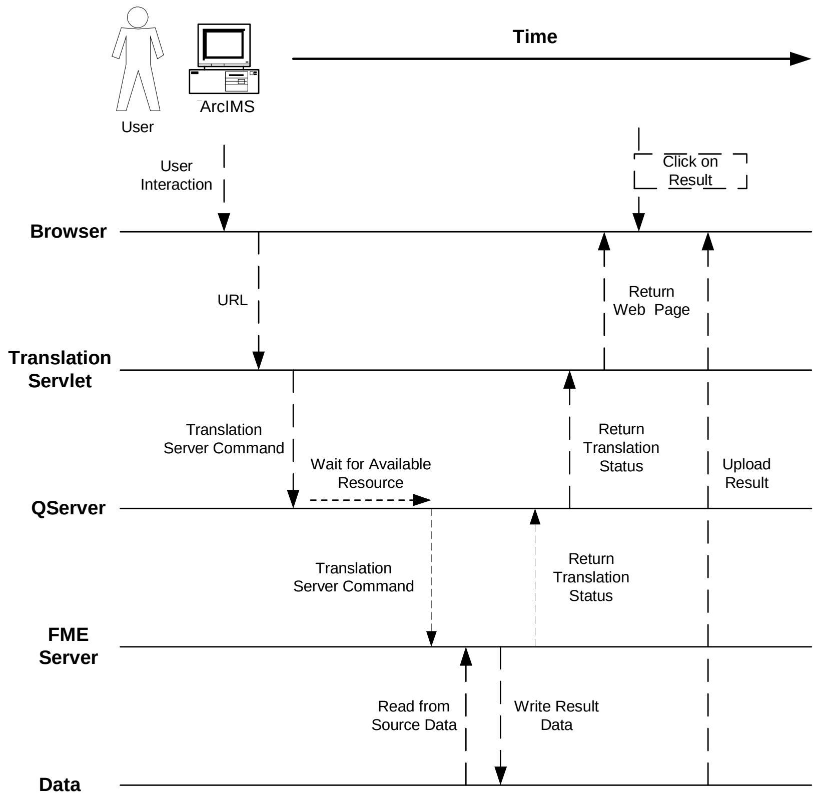


FIGURE 1-5 **DDE Sequence of Events**

DDE Start-up Options

Although it is possible to manually start each component of DDE individually, the usual and recommended procedure is to start the Process Monitor and have it automatically start up all the other components in sequence. The Process Monitor itself is started from a batch or script file.

The default distribution of DDE uses this start-up approach, which is illustrated below. This sequence would occur when starting DDE on a single system, using one QServer, one FME Server, and Sun's ServletRunner as the servlet engine. Windows batch files are shown first, followed by the equivalent UNIX script file in brackets where applicable:

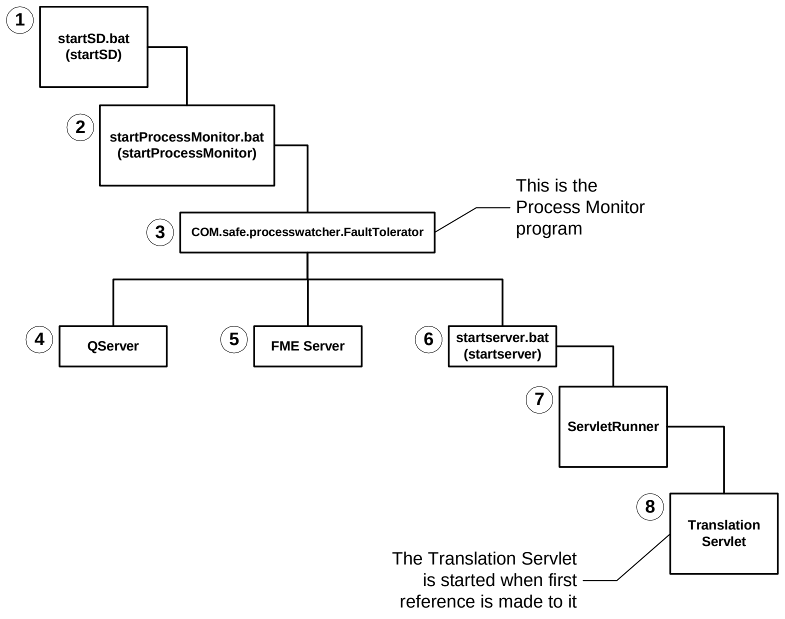


FIGURE 1-6 **Start-up Sequence**

Changes in the type, number and order of component startup are best made by modifying the `<DDEInstallDir>\processMonitorConfig.txt` file which specifies to the Process Monitor which components to run. This file is described in *Process Monitor Configuration* on page 234.

As shown in the figure above, running the `startSD` file initiates the start-up sequence. This file can be run manually from an interactive command line, or, on UNIX systems, automatically at system boot time.

To have DDE automatically start at system boot time on Windows systems, the Process Monitor can be run as a Windows Service. During DDE installation on Windows, the Process Monitor is installed (but not started) as a Windows Service and set to use the `<DDEInstallDir>\processMonitorConfig.txt` file.

DDE as a Windows Service on page 275 describes the procedure required to start and stop the Process Monitor as a Service.

CHAPTER 2

Getting Started

System Requirements

Windows

The minimum system requirements necessary to run DDE on Windows® systems are:

- a Pentium-based PC
- at least 256 MB of RAM
- a hard disk with at least 370 MB of free space – this does not include the space you require to hold the source and destination datasets
- Service Pack 4 or higher if running Windows NT.

UNIX

The minimum system requirements necessary to run DDE on a UNIX® system are:

- One of the following UNIX operating systems:
  - Sun® SPARC Solaris 2.x
  - IBM® RS/6000 AIX 4
  - Red Hat® Linux 6.2 Intel
  - Hewlett-Packard HP-UX 11.00
- at least 256 MB of RAM
- a hard disk with at least 370 MB of free space – this does not include the space you require to hold the source and destination datasets

For increased performance the available RAM should be increased. In addition, the system's I/O speed will affect translation throughput.

Software and Network Requirements

The following are the software and network requirements specific to DDE components:

- The system and supporting network on which DDE is installed must support the TCP/IP protocol.
- Each system on which DDE Java components are installed needs to support Sun's Java Runtime Environment Version 1.4.1 or higher. This is included with DDE installation.
- Each system on which DDE Java components are installed needs to support Sun's Java Servlet Development Kit 2.1 or higher, or Jakarta Tomcat. Both of these are included with DDE installation.
- A general-purpose web server must be running in the DDE environment to allow users to initiate access the system and to download translation results. DDE will work in conjunction with any standard web server. DDE's Translation Servlet must be able to read from the web server's public HTML directory (or an alias) and the FME Server(s) must be able to read/write from/ to this directory (or an alias).
- Basic servlet engine components are included with DDE installation in order to run the Translation Servlet. This servlet engine will not conflict with any existing installed servlet engines on the host system and is the only supported configuration of DDE. Knowledge Base articles describing the configuration of DDE with ArcIMS supported web server/servlet engine configurations are available for those who wish to customize their DDE installation.
- A web browser, such as Microsoft Internet Explorer or Netscape, is required for client interaction.
- ArcIMS 9.0 must be installed.

This reference manual assumes that the system administrator/DDE installer is familiar with the use and configuration of their web server and related internet technologies, with the installation and configuration of ArcIMS and with general geographic information systems (GIS) principles.

Licensing Requirements

DDE requires a separate license in addition to any existing ArcIMS license.

A single DDE license enables DDE (the FME Server component) on a single CPU and multiple licenses may be purchased to enable DDE for multiple CPUs.

DDE does not have to be installed on the same machine as ArcIMS.

DDE Installation Overview

In general, DDE installation is straightforward but it does require a step-by-step approach to handle all the details. Most of the effort with DDE occurs up-front with the installation and configuration of the product. Once the site-specific setup has been completed, DDE requires minimal maintenance (unless additional changes are made later on), and for the most part, it runs automatically in the background.

Below is an overview of the steps to follow when you're installing the ArcIMS DDE:

1 **Install ArcIMS.**

Ensure that you have a properly installed ArcIMS system running and correctly accessing and displaying your source data.

2 **Install DDE.**

Both Windows and UNIX installation instructions are given in *Installing and Running DDE* on page 19. UNIX installation requires editing a file to set parameter values and running a configuration script to perform the initial installation.

Installing DDE does not initially integrate it with ArcIMS. DDE supplies its own basic browser-oriented viewer allowing it to be used stand-alone for verification purposes. DDE also comes with sample source data and this is used to verify that the basic installation is working by displaying and translating the sample dataset in the browser viewer.

Before continuing You should make sure DDE is installed and successfully displaying and translating its own sample data.

3 **Configure DDE to read your own source data.**

This is described in *Configuring DDE to Read User Source Data* on page 59.

The steps in configuring DDE to read your source data mostly involve editing the content of several files in different locations. Basically you're telling DDE where your source data is, how to get to it, what format the source data is in, its coordinate system, etc.

Once all of this is completed, you should be able to use DDE (which is still stand-alone and not yet integrated with ArcIMS) to display and translate your own source dataset.

Before continuing You should make sure DDE can successfully access your own source data.

4 **Install the additional software that integrates DDE with ArcIMS.**

The software to do this is in the `DDEIntegration.zip` file included on the CD. Part 2 of this manual describes the necessary steps required to integrate DDE with ArcIMS.

The integration procedure involves adding and modifying a handful of files to your existing ArcIMS installation and to your DDE installation. Essentially what you're doing is telling both systems about each other and mapping ArcIMS's definition of the source data to that used by DDE.

It's important to realize that ArcIMS and DDE each read your source data independent of each other. DDE doesn't "go through" ArcIMS to access your source data, and each system has its own independent view of the source data. Much of the integration procedure involves making sure that both systems agree in their respective views of the source data.

Once the integration is completed successfully, your ArcIMS viewer should have a new toolbar button enabling you to connect to the DDE front-end, submit a translation request and download the translation results.

CHAPTER

3

Installing and Running DDE

This chapter includes detailed Windows and UNIX installation procedures.

If you're installing DDE on Windows, please refer to the following sections:

- *Installing DDE on a Single Windows System*: The screen captures illustrate the sequence of interactive panels that will guide you through a standard default installation of DDE on a single Windows target system.
- *Starting DDE on Windows*: How to start DDE interactively as part of the installation and testing process. After initial installation and configuration, DDE can be set up to start as a service.
- *Logging into DDE on Windows*: How to log into DDE to confirm basic operation for a standalone DDE system, prior to integration with ArcIMS.

Additional sections give instructions on shutting down DDE on Windows, and uninstalling DDE from Windows.

If you're installing DDE on UNIX, please refer to the following sections:

- *Installing DDE on a Single UNIX System*: Numbered steps guide you through a standard default installation of DDE on a single UNIX target system.
- *Starting DDE on UNIX*: How to start DDE as part of the installation and testing process.
- *Logging into DDE on UNIX*: How to log into DDE to confirm basic operation for a standalone DDE system, prior to integration with ArcIMS.
- Certain patches are sometimes required on UNIX systems – you may also have to refer to the sections that reference these patches.

Additional sections give instructions on shutting down DDE on UNIX, and uninstalling DDE from UNIX.

Installing DDE on a Single Windows System

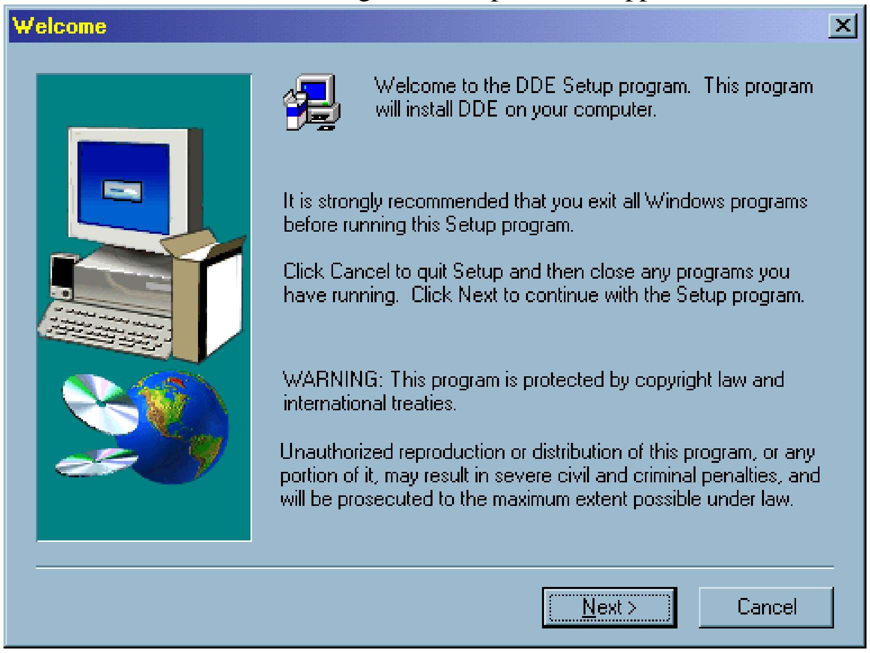
The following sections describe the installation of DDE on a single Windows system. It is highly recommended that you do the following before you install DDE on a Windows platform:

- Exit all Windows programs to free up memory and prevent possible conflicts between the Installer and other programs.
- Install the software from a user account that has Administrator privileges.

DDE includes all component software necessary to get the product up and running. DDE is available either on a CD or as a downloadable compressed `.zip` file. The installation steps are described below.

1 Installation Welcome Panel

To start the DDE installation procedure, double-click the `DDE.exe` installation file. The following welcome panel will appear:



Welcome to the DDE Setup program. This program will install DDE on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup program.

Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.

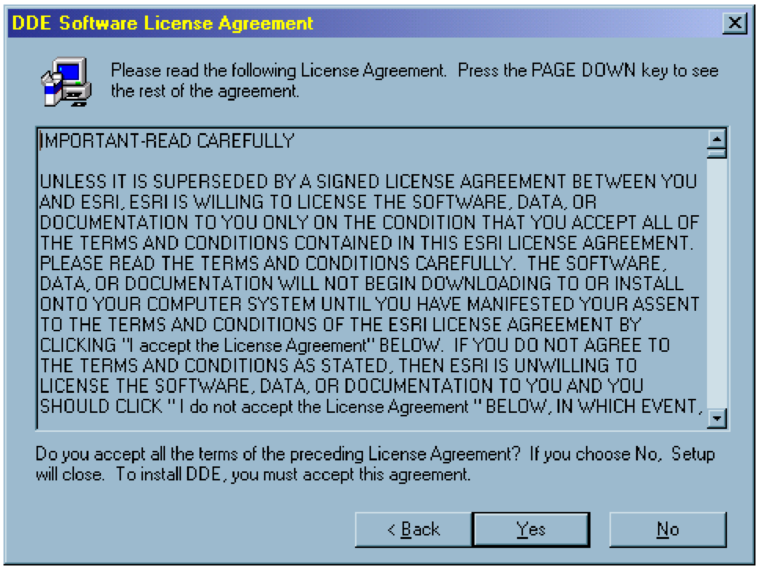
WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

Click **Next** to continue.

2 License Panel

Review the scrollable license agreement text in the license panel. If you agree with terms, click **Yes** to continue.



DDE Software License Agreement

Please read the following License Agreement. Press the PAGE DOWN key to see the rest of the agreement.

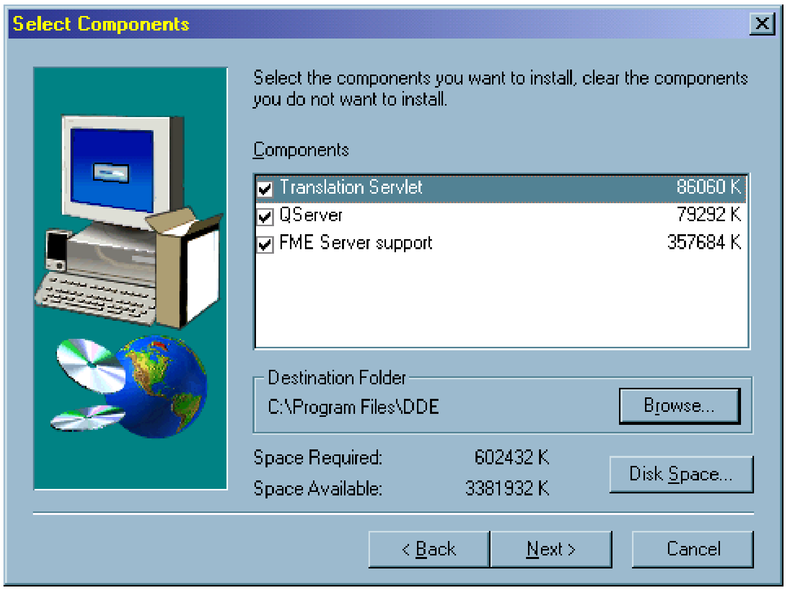
IMPORTANT-READ CAREFULLY

UNLESS IT IS SUPERSEDED BY A SIGNED LICENSE AGREEMENT BETWEEN YOU AND ESRI, ESRI IS WILLING TO LICENSE THE SOFTWARE, DATA, OR DOCUMENTATION TO YOU ONLY ON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS AND CONDITIONS CONTAINED IN THIS ESRI LICENSE AGREEMENT. PLEASE READ THE TERMS AND CONDITIONS CAREFULLY. THE SOFTWARE, DATA, OR DOCUMENTATION WILL NOT BEGIN DOWNLOADING TO OR INSTALL ONTO YOUR COMPUTER SYSTEM UNTIL YOU HAVE MANIFESTED YOUR ASSENT TO THE TERMS AND CONDITIONS OF THE ESRI LICENSE AGREEMENT BY CLICKING "I accept the License Agreement" BELOW. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS AS STATED, THEN ESRI IS UNWILLING TO LICENSE THE SOFTWARE, DATA, OR DOCUMENTATION TO YOU AND YOU SHOULD CLICK " I do not accept the License Agreement " BELOW, IN WHICH EVENT,

Do you accept all the terms of the preceding License Agreement? If you choose No, Setup will close. To install DDE, you must accept this agreement.

3 Component Selection Panel

Accept the default component selection that specifies all DDE components for installation. Then either browse to and select the desired directory into which DDE is to be installed, or accept the default location. Please make note of the disk space required and available.



Select Components

Select the components you want to install, clear the components you do not want to install.

Components

✓ Translation Servlet	86060 K
✓ QServer	79292 K
✓ FME Server support	357684 K

Destination Folder
C:\Program Files\DDE

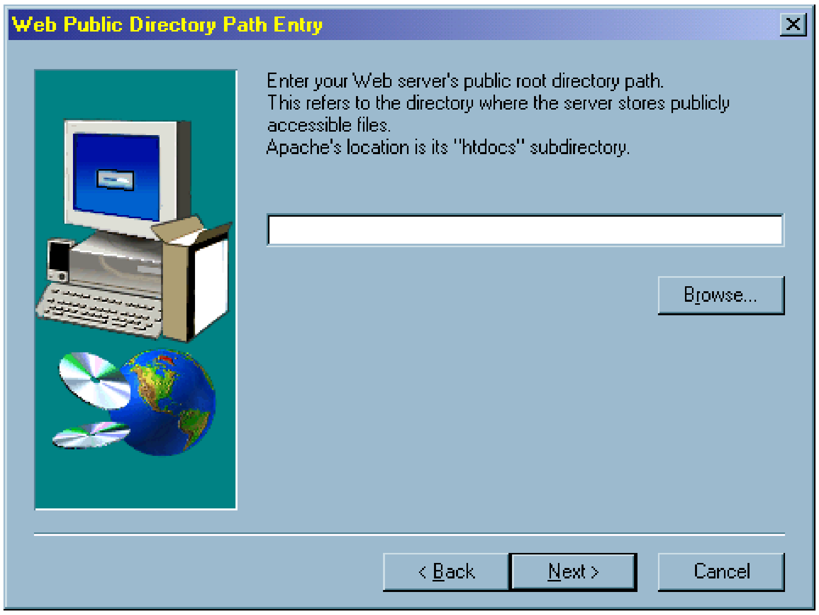
Space Required: 602432 K
Space Available: 3381932 K

Click **Next** to continue.

4 Webserver Public Directory Panel

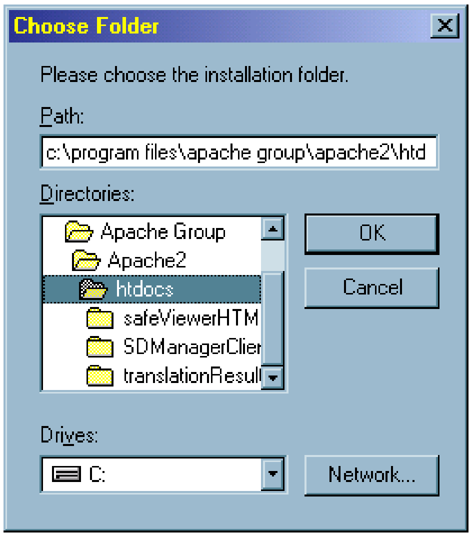
Either enter or browse to and select the filepath to the web server's public root directory. This is the directory in which the web server being used with DDE stores publicly accessible files. The location depends on the web server in use. For example, the default location used by the Apache web server is `C:\ProgramFiles\ApacheGroup\Apache2\htdocs`. The equivalent location for the IIS web server is `C:\Inetpub\wwwroot`.

The panel initially looks like this:



Enter your Web server's public root directory path.
This refers to the directory where the server stores publicly accessible files.
Apache's location is its "htdocs" subdirectory.

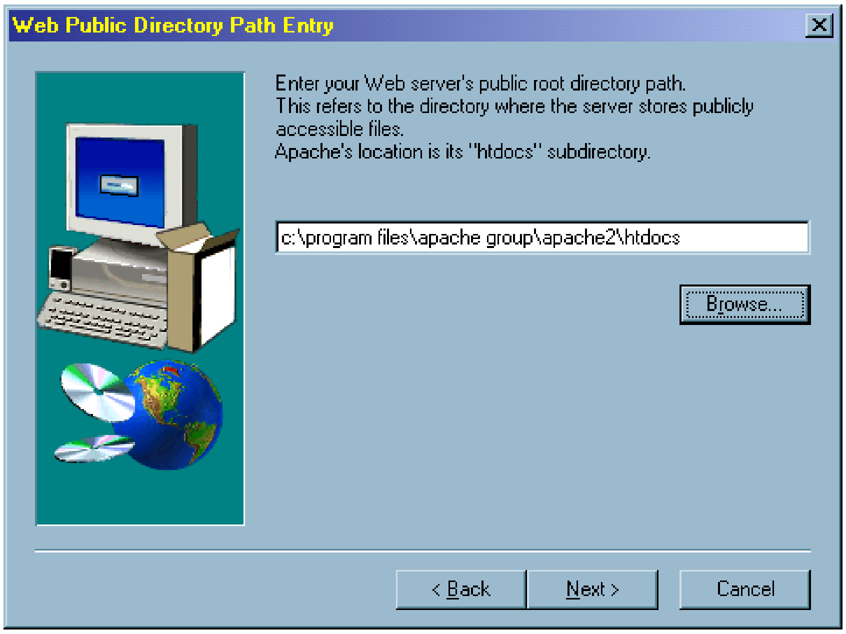
In the example below, the user has clicked the **Browse...** button and browsed to the default Apache public documents directory `htdocs`



Please choose the installation folder.

Path: c:\program files\apache group\apache2\htd

Select this directory and click OK to enter the filepath into the main panel as shown here:



Web Public Directory Path Entry

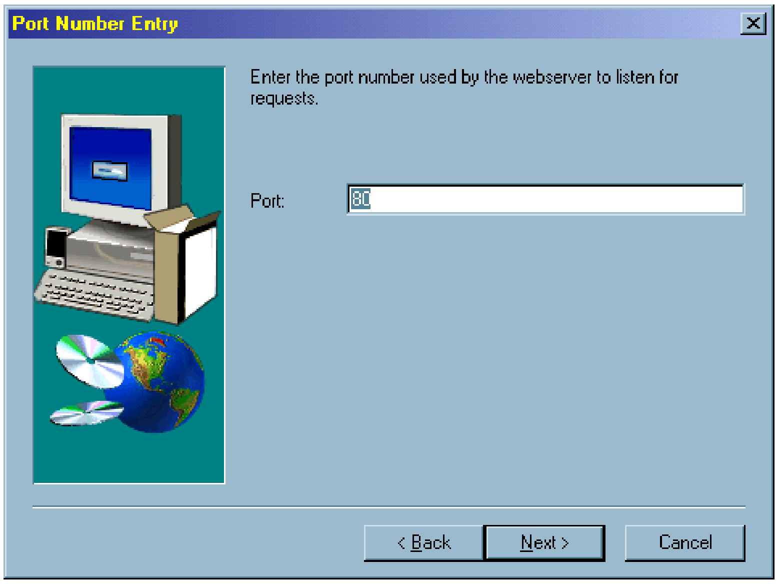
Enter your Web server's public root directory path.
This refers to the directory where the server stores publicly accessible files.
Apache's location is its "htdocs" subdirectory.

c:\program files\apache group\apache2\htdocs

Click **Next** to continue.

5 Web Server Port Panel

Enter the port number on which the web server listens for requests. The supplied default is port 80, which is the standard default for most web servers and can be accepted as-is in virtually all cases.



Port Number Entry

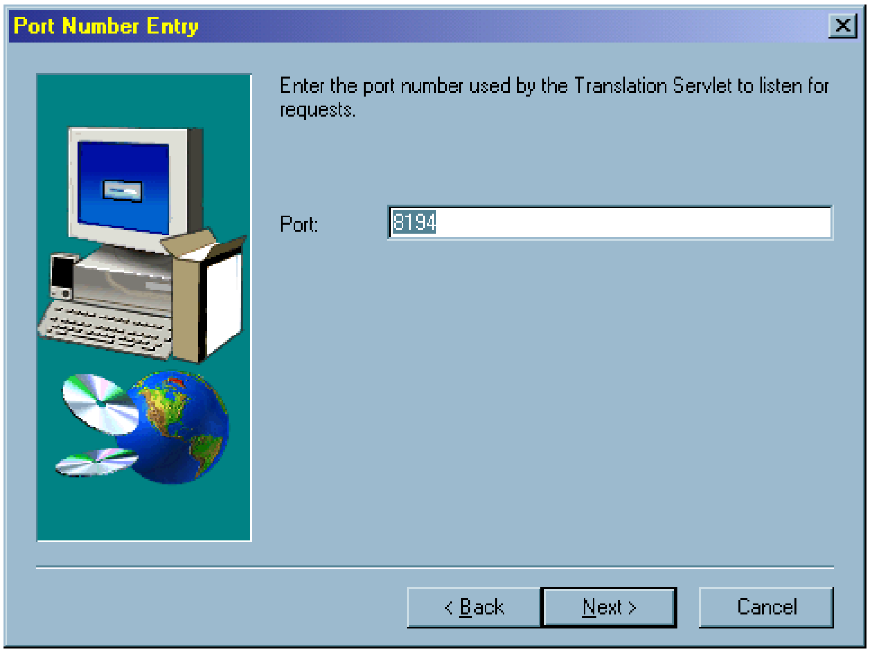
Enter the port number used by the webserver to listen for requests.

Port: 80

Click **Next** to continue.

6 Servlet Engine Port Panel

Enter the port number on which the DDE Translation Servlet servlet engine (Tomcat) listens for translation requests. The supplied default is port 8194, and should be accepted unless it is known that this port is already in use by another application running on the same system. In this case, a different, unused port number should be entered here. Any port number can be specified, as long as it isn't already in use.



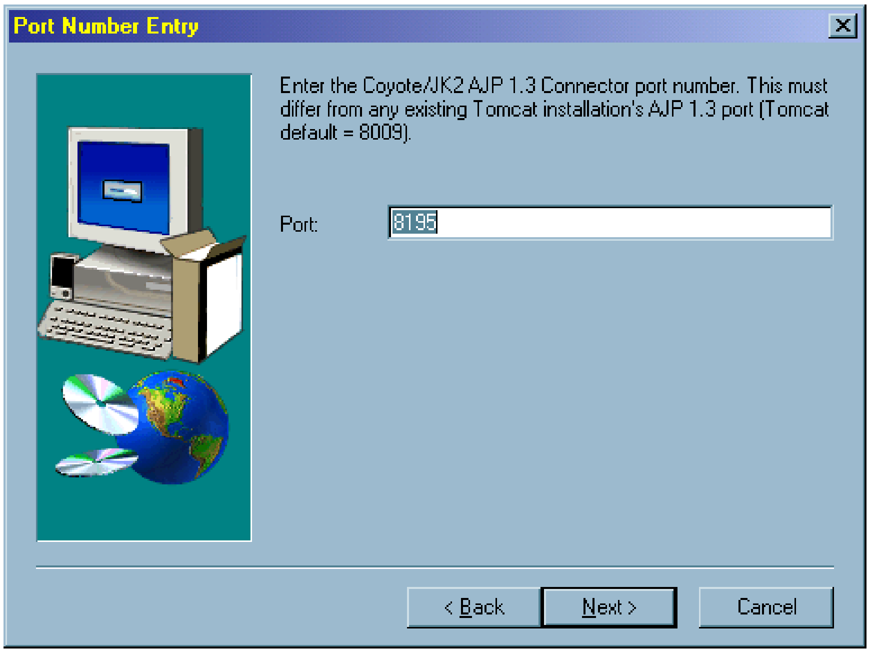
Enter the port number used by the Translation Servlet to listen for requests.

Port: 8194

Click **Next** to continue.

7 AJP Connector Port Panel

Enter the port number which the DDE Translation Servlet servlet engine (Tomcat) uses for its AJP Connector. The supplied default is port 8195, and should be accepted unless it is known that this port is already in use by another application running on the same system. In this case a different, unused port number should be entered here. Any port number can be specified, as long as it isn't already in use.



Enter the Coyote/JK2 AJP 1.3 Connector port number. This must differ from any existing Tomcat installation's AJP 1.3 port (Tomcat default = 8009).

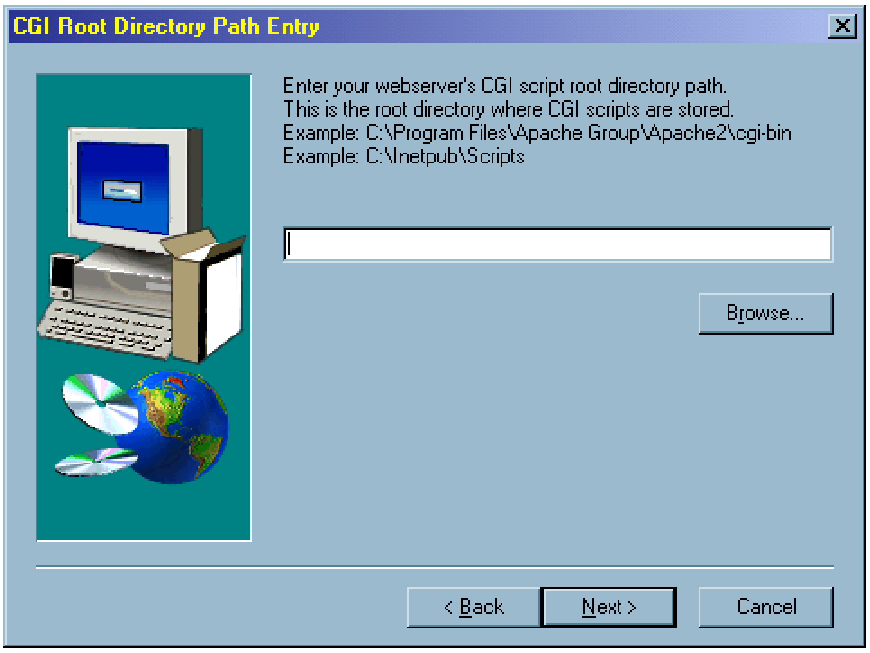
Port: 8195

Click **Next** to continue.

8 CGI Script Directory Panel

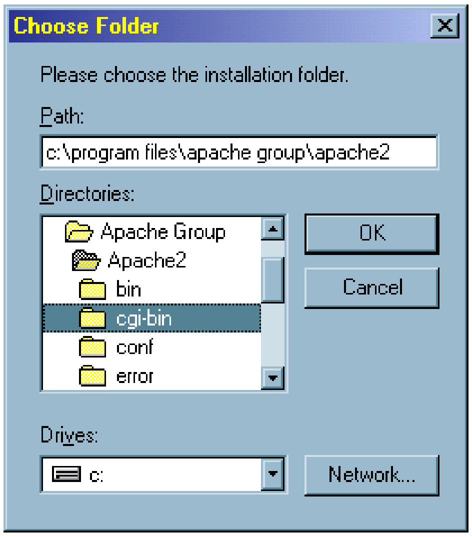
Either enter or browse to and select the filepath to the web server's CGI script root directory. This is the directory in which the web server stores CGI script files. The location depends on the web server being used. For example, the default location used by the Apache web server is `C:\Program Files\Apache Group\Apache2\cgi-bin`. The equivalent location for the IIS web server is `C:\Inetpub\Scripts`.

The panel looks initially like this:



Enter your webserver's CGI script root directory path.
This is the root directory where CGI scripts are stored.
Example: C:\Program Files\Apache Group\Apache2\cgi-bin
Example: C:\Inetpub\Scripts

In the example below, the user has clicked the **Browse...** button and browsed to the default Apache CGI script root directory `cgi-bin`



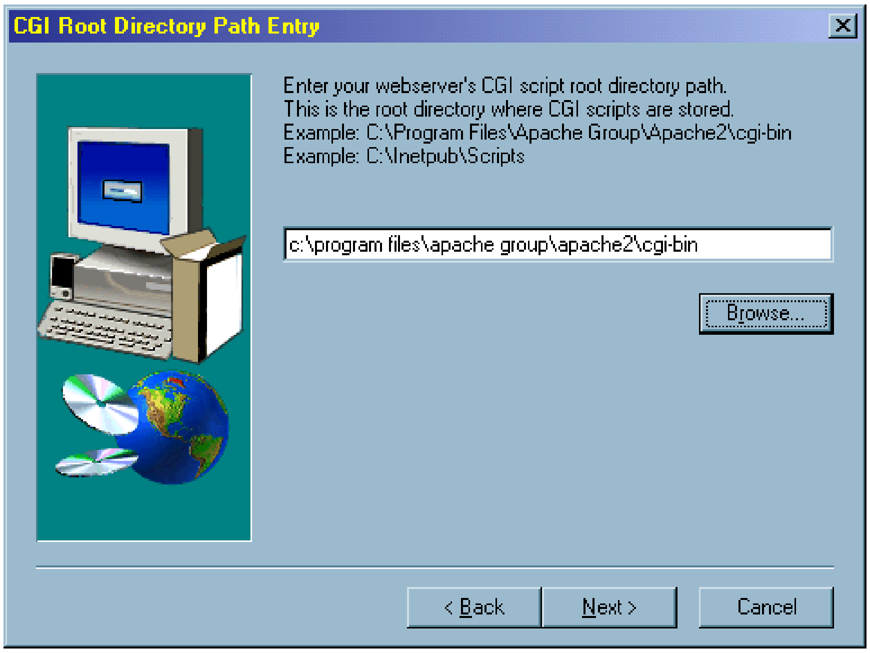
Please choose the installation folder.

Path: c:\program files\apache group\apache2

Directories:
- Apache Group
- Apache2
- bin
- cgi-bin
- conf
- error

Drives: c:

Select this directory and click OK to enter the filepath into the main panel as shown below:



CGI Root Directory Path Entry

Enter your webserver's CGI script root directory path.
This is the root directory where CGI scripts are stored.
Example: C:\Program Files\Apache Group\Apache2\cgi-bin
Example: C:\Inetpub\Scripts

c:\program files\apache group\apache2\cgi-bin

Click **Next** to continue.

9 CGI Script Location Names Panel

a. In the "URL name" field, enter the name that is used within URLs to reference the web server's CGI script root directory. This name is part of a URL - it is not the same as the filepath to the directory itself (entered in the previous step). For example, on a system named "green" running the standard Apache web server, the actual filepath to the CGI script root directory is:

```
C:\Program Files\Apache Group\Apache2\cgi-bin
```

whereas the URL name referencing the same root directory is (by default) "cgi-bin". In use, the URL to this directory on system "green" would then be:

```
http://green/cgi-bin
```

Similarly, on a system named "red" running the standard IIS web server, the actual filepath to the CGI script root directory is:

```
C:\Inetpub\Scripts
```

whereas the URL name referencing the same root directory is (by default) "Scripts". In use, the URL to this directory on system "red" would then be:

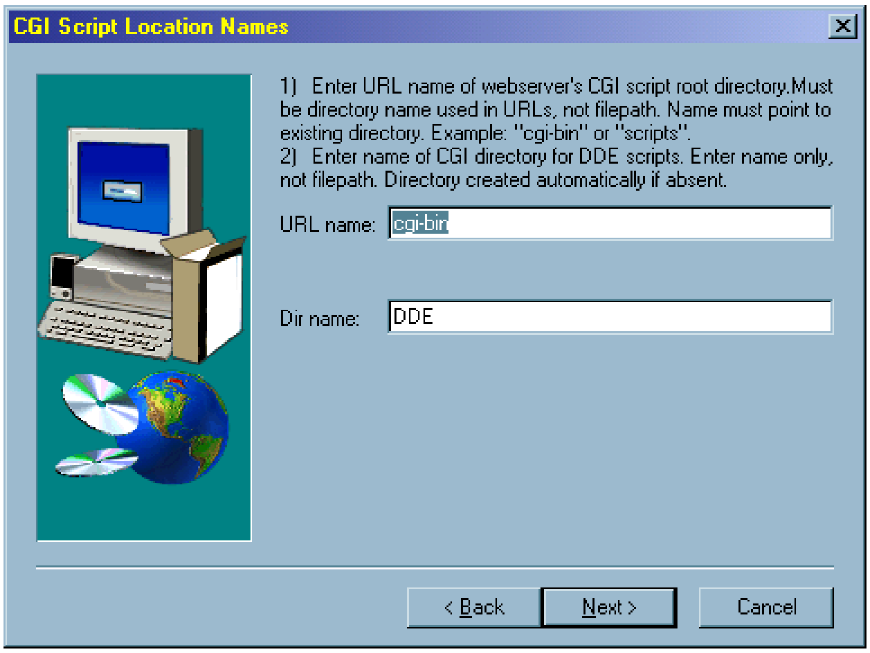
```
http://red/Scripts
```

Note: The default IIS web server installation does not automatically create an alias to the Scripts directory. It may be necessary to create a virtual directory using the IIS administration tool to point to the physical Scripts directory so that the URL *http://<webservername>/ Scripts* (for example, *http://red/Scripts*) is a valid URL.

It is the directory alias portion of the URL name and not the directory filepath that must be entered into the “URL name” field of the panel.

b. In the “Dir name” field, enter the name of a subdirectory that will be automatically created *within* the CGI script root directory to hold the DDE-specific CGI script files. Enter a name only and not a full filepath.

The example panel below shows the supplied default values in effect when the Apache web server is being used.



1) Enter URL name of webserver's CGI script root directory.Must be directory name used in URLs, not filepath. Name must point to existing directory. Example: "cgi-bin" or "scripts".
2) Enter name of CGI directory for DDE scripts. Enter name only, not filepath. Directory created automatically if absent.

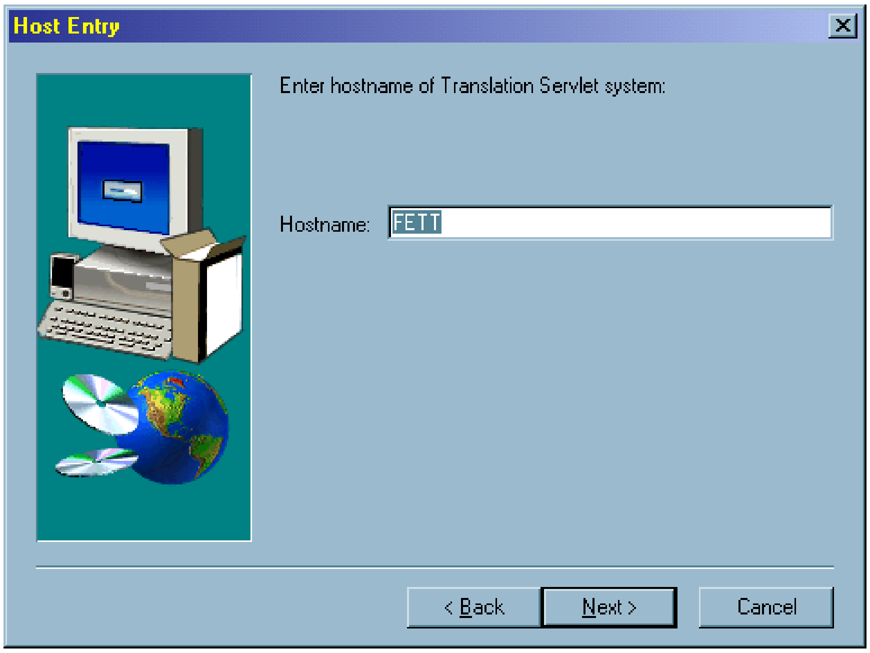
URL name: cgi-bin

Dir name: DDE

Click **Next** to continue.

10 Translation Servlet Hostname Panel

Enter the hostname of the system on which the DDE Translation Servlet component is running. The supplied default value will already be set to the name of the system on which DDE is being installed and should be accepted as-is.


Host Entry

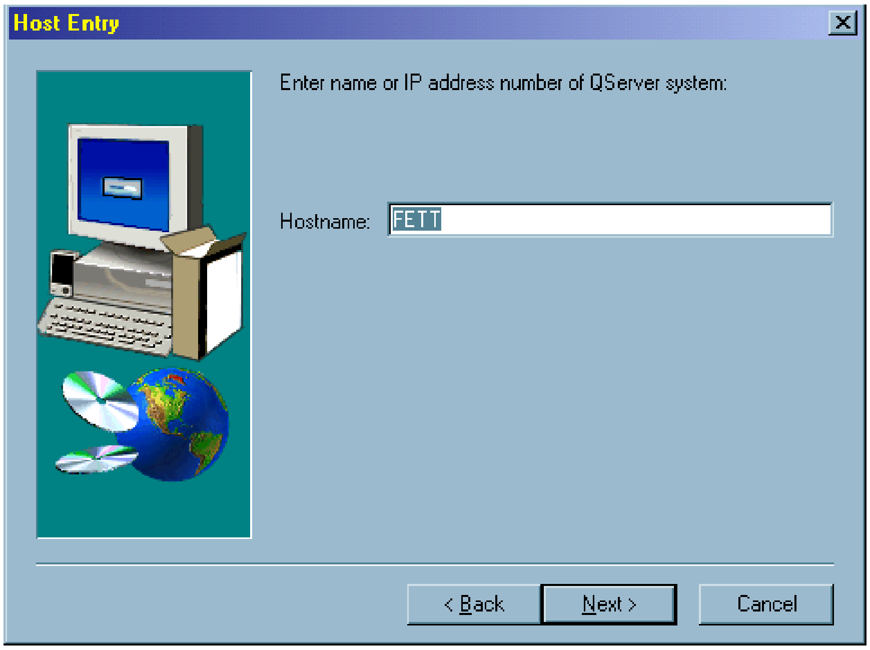
Enter hostname of Translation Servlet system:

Hostname: FETT

Click **Next** to continue.

11 QServer Hostname Panel

Enter the hostname of the system on which the DDE QServer component is running. The supplied default value will already be set to the name of the system on which DDE is being installed and should be accepted as-is. Alternatively, you can enter the full, numeric IP address of the system instead of the name. This alternative is typically used only when the QServer component is installed separately on a system of its own and is running from behind a firewall. When installing DDE on a single system, you should specify the name, not the IP address.



Enter name or IP address number of QServer system:

Hostname: FETT

Click **Next** to continue.

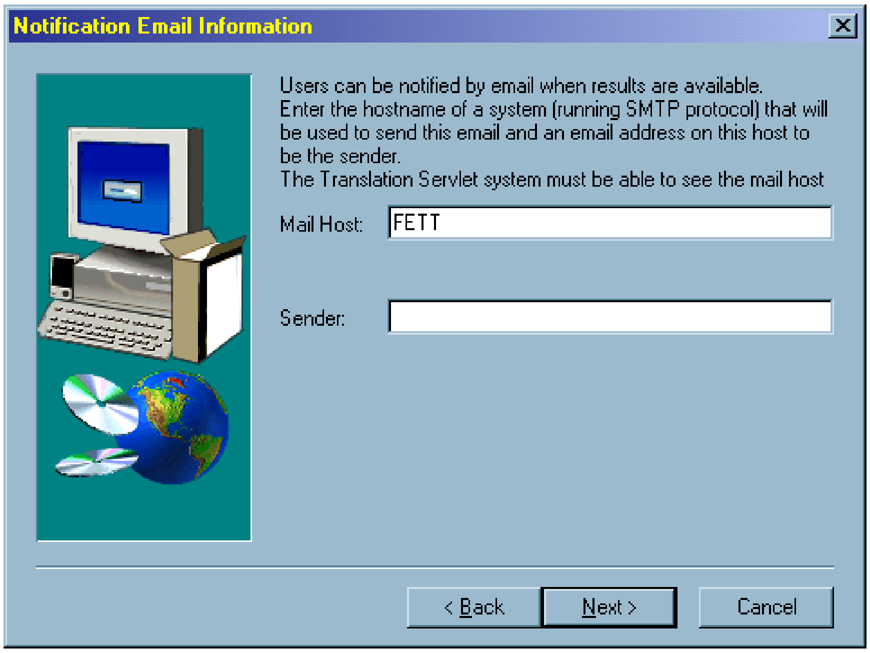
12 Notification Email Panel

a. In the "Mail Host" field, enter the hostname of a system running an SMTP mail server. DDE will use the SMTP mail server running on this system to send e-mail notifications to end users if so configured. E-mail notification is optional and is disabled by default. Enabling and using notification e-mail is described in *Result Notification via E-mail* on page 228.

The system on which DDE is being installed must be able to access and use this SMTP server. The supplied default value will already be set to the name of the system on which DDE is being installed. This system can be used as long as it has an SMTP server running on it. If not, either specify the name of another system that is running an accessible SMTP server, or, if none is available, simply use the default as a dummy value.

When an SMTP server is not available, DDE will not send e-mail notifications. This is not an error, and if an SMTP server becomes subsequently available, DDE parameters can be readjusted to activate e-mail notification using this server.

In the example panel below, the SMTP mail server has been set to be "FETT".



Notification Email Information

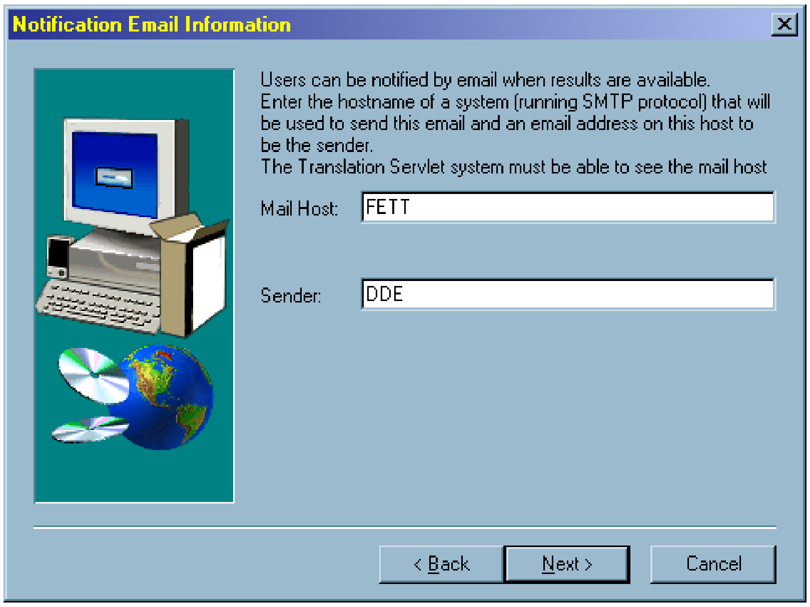
Users can be notified by email when results are available.
Enter the hostname of a system (running SMTP protocol) that will be used to send this email and an email address on this host to be the sender.
The Translation Servlet system must be able to see the mail host

Mail Host: FETT

Sender:

b. In the “Sender” field, enter the name to be used as the sender of DDE e-mail notifications. End users receiving e-mail notifications from DDE will see this name as the e-mail sender.

The sender name can be the name of an existing e-mail account known to the SMTP server. However, when enabled, DDE notifications only *send* email, they never receive any, so any descriptive name can be entered here. It can, but does not need to be, the name of an existing e-mail account on the SMTP server. In the example panel below, the sender name “DDE” has been entered into the “Sender” field.



Notification Email Information

Users can be notified by email when results are available.
Enter the hostname of a system (running SMTP protocol) that will be used to send this email and an email address on this host to be the sender.
The Translation Servlet system must be able to see the mail host

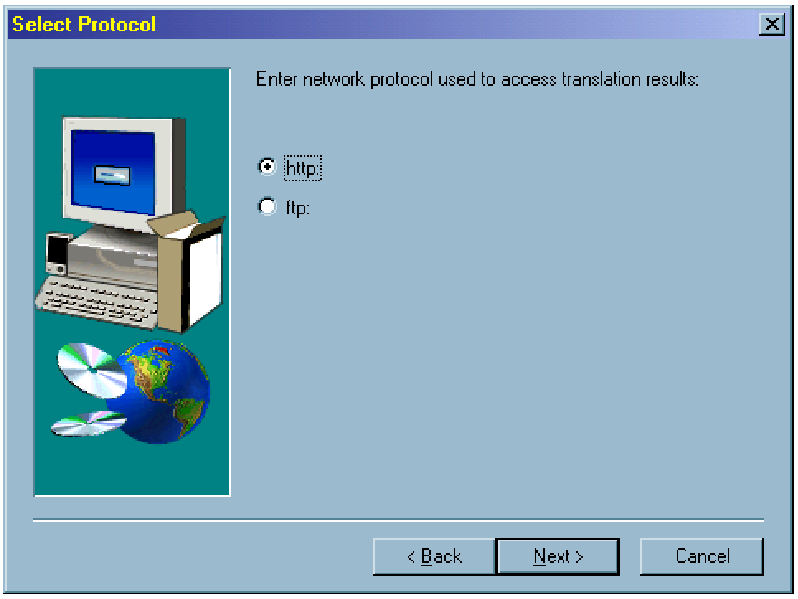
Mail Host: FETT

Sender: DDE

Click **Next** to continue.

13 Results Access Protocol

Select the access protocol that DDE should use in accessing translation results. Accept the supplied default of “http:” unless an FTP Server has been specifically set up to provide this service for DDE, in which case select “ftp:”. The vast majority of DDE installations will use the “http:” default.



Select Protocol

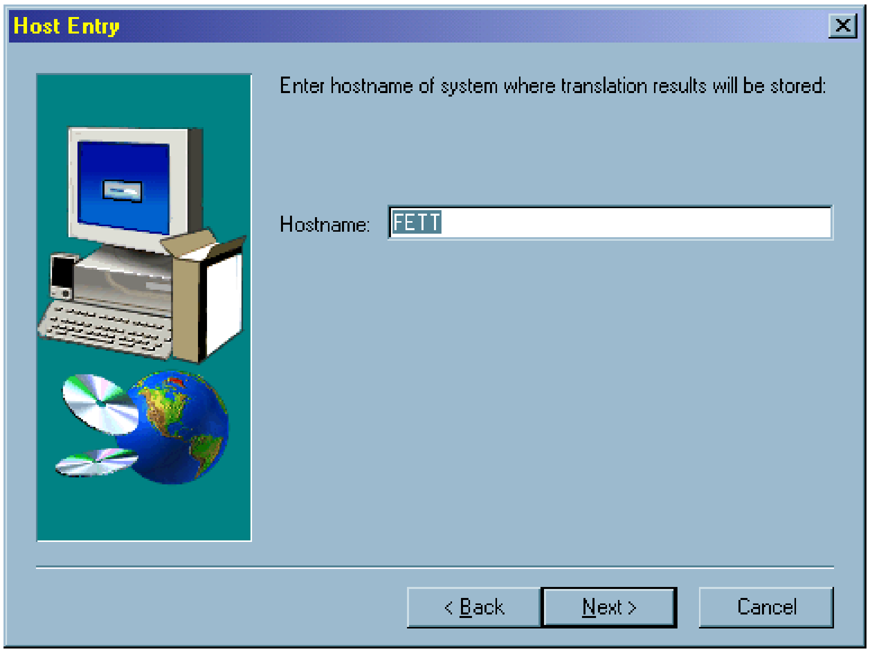
Enter network protocol used to access translation results:

[✓ http:]
ftp:

Click **Next** to continue.

14 Translation Results Hostname Panel

Enter the hostname of the system on which the DDE is to store translation result files. The supplied default value will already be set to the name of the system on which DDE is being installed and should be used as-is.



Host Entry

Enter hostname of system where translation results will be stored:

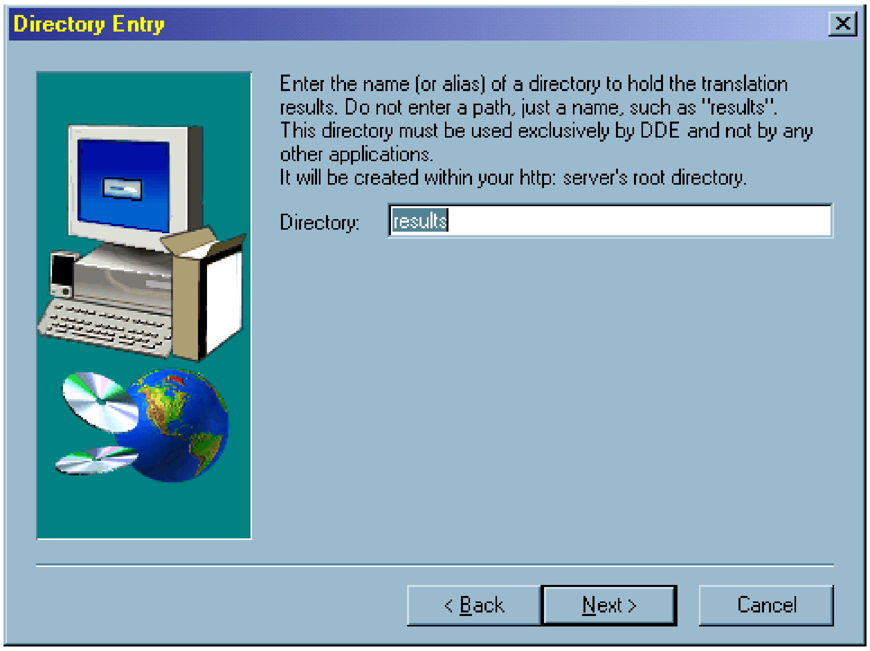
Hostname: FETT

Click **Next** to continue.

15 Translation Results Directory Panel

Enter the name of a directory into which DDE will store translation results. This directory will be located within the web server's public root directory (specified in Step 4 above) and will be created in that location automatically if it doesn't already exist there. This directory is for the exclusive use of DDE and should not be used by any other application. The contents of this directory will be periodically deleted by DDE as a cleanup measure during the course of normal operations.

Enter only a single name such as "results" or "translationOutput", not a full directory filepath. In the example panel below, the supplied default name of "results" has been used.



Enter the name (or alias) of a directory to hold the translation results. Do not enter a path, just a name, such as "results". This directory must be used exclusively by DDE and not by any other applications.
It will be created within your http: server's root directory.

Directory: results

Click **Next** to continue.

16 DDE Administration Panel

DDE accepts administration commands sent to it programmatically, most notably the command to shut itself down. The following parameters configure DDE for accepting these commands.

a. In the "Admin Port" field, enter the port number on which DDE listens for administration requests. The supplied default is port 7500 and should be accepted unless it is known that this port is already in use by another application running on the same system. In this case a different, unused port number should be entered here. Any port number can be specified, as long as it is not already in use.

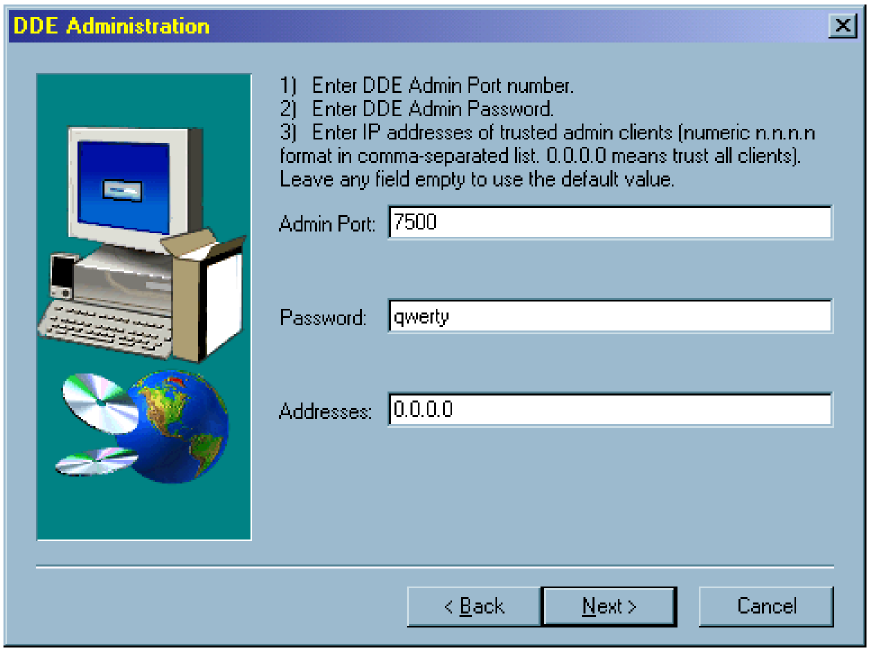
b. In the "Password" field, enter a password string that will be used by DDE to validate administration requests sent to it. The supplied default value is initially set to a random number which may be changed to something more easily remembered.

c. In the "Addresses" field, enter the IP addresses of client systems that DDE should trust and therefore accept administration requests from (chiefly the request to shut down). Each address must be specified in numeric `n.n.n.n` format. If more than one address is specified, each should be separated from the other by a comma. Here is an example specifying three trusted client IP addresses:

```
138.45.2.12,163.281.8.2,123.219.72.12
```

The supplied default value of “0.0.0.0” is a wildcard value which means all clients are trusted regardless of their IP address. This is the recommended value to use for the initial DDE installation. Once DDE is installed, specific IP addresses can be subsequently defined in the DDE configuration as necessary, once they have been determined.

In the example panel below, the default admin port and wildcard trusted address values have been used, and the admin password has been changed to “qwerty”.



1) Enter DDE Admin Port number.
2) Enter DDE Admin Password.
3) Enter IP addresses of trusted admin clients (numeric n.n.n.n format in comma-separated list. 0.0.0.0 means trust all clients).
Leave any field empty to use the default value.

Admin Port: 7500

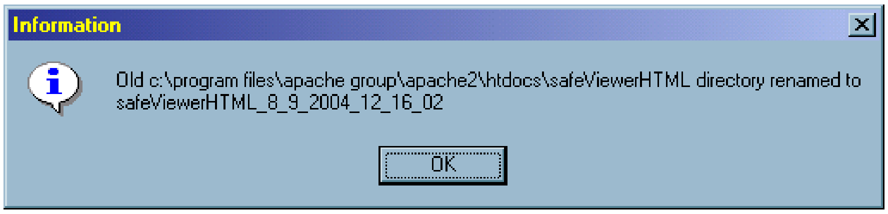
Password: qwerty

Addresses: 0.0.0.0

Click **Next** to continue.

17 Previous Files Renamed Panel(s)

If a version of DDE was installed previously and was never uninstalled, and the current installation is being done in the same location, one or more information panels may appear indicating that previous configuration data has been renamed in order to preserve it. These panels will look similar to this:

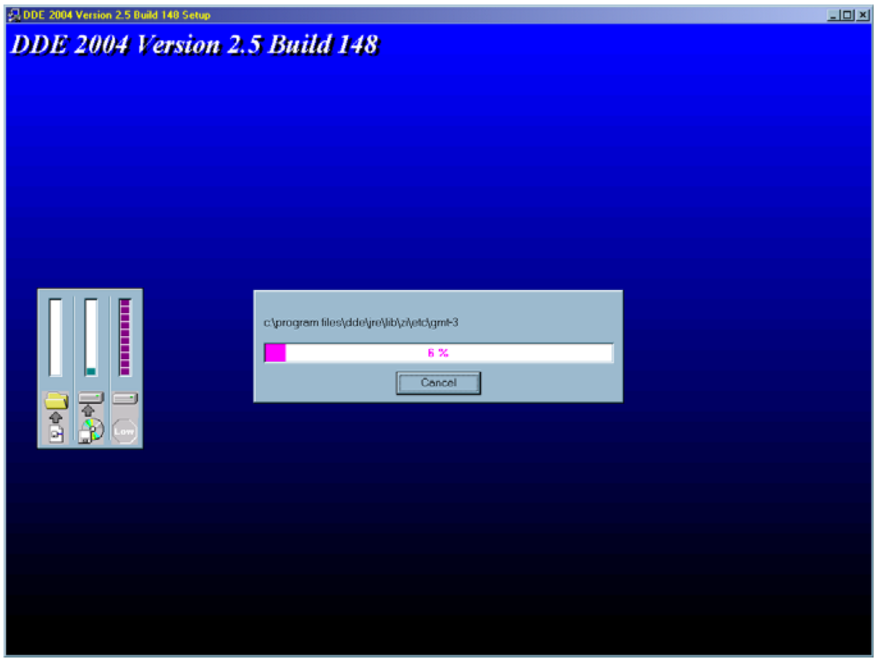


Old c:\program files\apache group\apache2\htdocs\safeViewerHTML directory renamed to safeViewerHTML_8_9_2004_12_16_02

If any of these panels appear, click **OK** to continue.

18 Installation File Copying and Configuration

The DDE installation will display a progress bar as files are copied to the target installation directory. The installation screen will look similar to this:

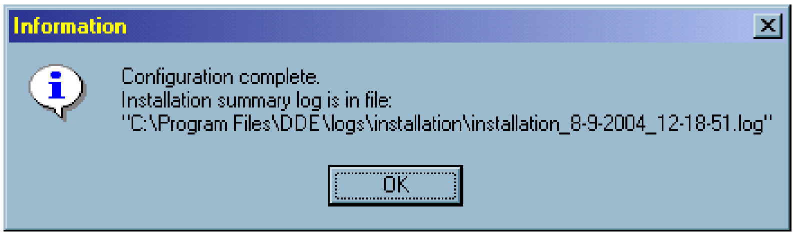

DDE 2004 Version 2.5 Build 148 Setup
DDE 2004 Version 2.5 Build 148
6 %
Cancel
Low

c:\program files\dde\jre\lib\zi\etc\gmt-3

At approximately the 50% point, a series of empty DOS windows will appear, alternating with beeping information panels indicating which DDE component has just been configured.

19 Configuration Completion Panel

Once all component configuration has been performed, the configuration completion panel is displayed, showing the filepath to a local text file containing a log of the DDE installation and configuration steps. This file can be examined to confirm that installation was successful, or as a debugging aid to assist in determining installation failures.

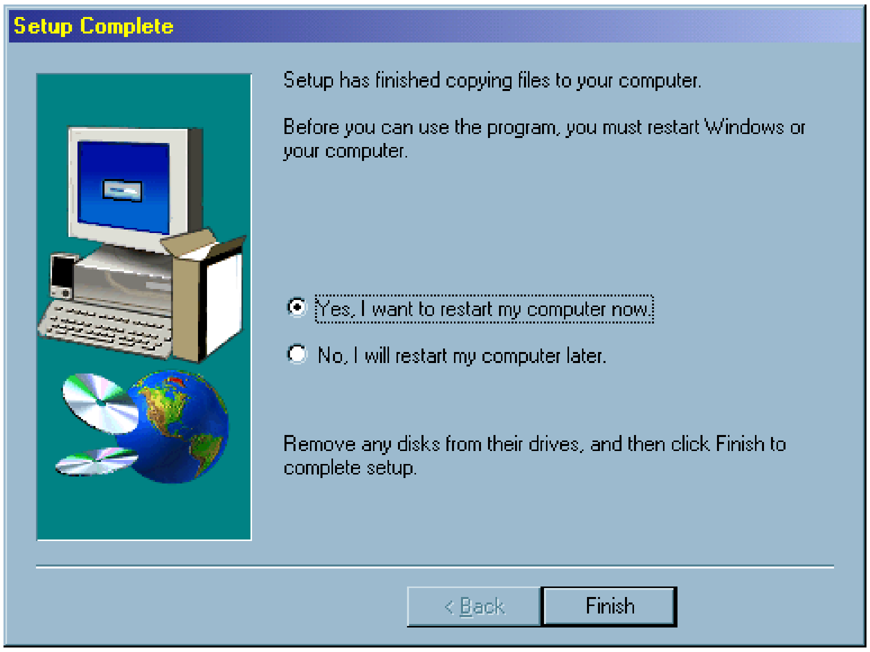


Configuration complete.
Installation summary log is in file:
"C:\Program Files\DDE\logs\installation\installation_8-9-2004_12-18-51.log"

Click **OK** to continue.

20 Setup Completion Panel

Finally, the setup completion panel is displayed, indicating that DDE installation has finished. You must restart the system before using DDE.


Setup Complete

Setup has finished copying files to your computer.

Before you can use the program, you must restart Windows or your computer.

[✓ Yes, I want to restart my computer now.]

No, I will restart my computer later.

Remove any disks from their drives, and then click Finish to complete setup.

Click **Finish** to complete the DDE installation.

Configuring IIS Web Servers After Rebooting

If the webserver being used is Microsoft Internet Information Services (IIS 5.0 or later), then it must be configured to run the CGI Perl script which forwards translation requests to the Translation Servlet component.

General documentation explaining the configuration of IIS to run CGI Perl scripts is contained in the online Microsoft documentation. To access the online documentation from the computer running your webserver, open an Internet browser window and go to the address `http://localhost/iishelp`.

The online help sections applicable to configuration of CGI Perl scripts are found under the contents titles:

```
Administration | Server Administration | Configuring
Applications | Configuring CGI Applications
Administration | Server Administration | Configuring
Applications | Setting Application Mappings
```

The online help system provides the most accurate and up-to-date instructions for configuring CGI Perl scripts to run on IIS 5.0 and should be used as the primary source of information for this task. The instructions given below provide a quick reference for implementing CGI Perl support for the translation request forwarding but, where differences arise, please refer to the online Microsoft help system.

1. Using the IIS administration tool, highlight the `cgi/scripts` directory created for scripts during the DDE installation process. (Typically this will be under **Default Web Site | Scripts | DDE** and it will contain the file `spatialDirect.pl`.)
2. Select Action | Properties from the IIS administration tool menu.
3. Select the tab entitled either Home Directory, Virtual Directory or Directory.
4. Under the Application Settings section click the Create button, type an Application Name (DDE is a good choice) and click the Configuration button.
5. Select the App Mappings tab and click the Add button to define the "Application" (that is, tell IIS which program is to be used to run Perl scripts).
6. Click the Browse button and select the `nsperl.exe` program which is located in the Perl directory within your DDE installation directory. Click the Open button.

   The full path to the `nsperl.exe` program should now be in the Executable box. To the end of this path, you now need to add

   ```
   %s %s
   ```

   The Executable box should now contain something similar to:

   ```
   "C:\Program Files\DDE\Perl\nsperl.exe" %s %s
   ```

   Note: Be sure to include the file path in quotation marks, as shown above, if the file path contains any embedded spaces. The %s %s string should be outside the quoted file path.
7. In the Extension box, type the extension

   ```
   .pl
   ```
8. Ensure that the Script Engine checkbox is checked and click OK to dismiss the window.
9. Click OK to dismiss the next two windows.

Testing Basic DDE Operation

At this point, installation of the basic DDE is complete and it is configured to work with the sample data provided with the application.

Test basic DDE operation by following these three steps:

- Start DDE (see *Starting DDE on Windows*).

Note: In order to run successfully, DDE must be licensed. Licensing is managed during the ArcIMS Post-Install with the Software Authorization Wizard (SAW). Your ArcIMS license file must include the "delivery" extension license; if "delivery" is not shown as being licensed upon completion of the SAW, then DDE will not run.

If your ArcIMS license does not include the Data Delivery Extension, then you can license DDE after ArcIMS has already been installed by obtaining a DDE license file and running the ArcIMS Software Authorization Wizard (SAW).

- Log into DDE (see *Logging into DDE on Windows*).
- Click on a grid square on the Index Map page to zoom in to that area.

If these three steps are successful, then you are ready to move on to configuring DDE to read user source data.

Replacing an Existing DDE Installation on Windows

If DDE is installed on top of an existing installation, it will replace all existing files with the files from the latest install package. However, the existing `translationControl`, `safeViewerHTML` and `JSDK webpages\Web-inf` directories are first backed up. The installation accomplishes this by renaming these directories by adding a timestamp suffix to their names before the new versions are installed. This preserves any custom user modifications made to the contents of these directories.

The prompt for the web server's public `root` directory path refers to the directory where the web server stores its publicly accessible files. For the Apache web server, this is the `htdocs` subdirectory whose pathname in the default Apache installation is:

```
C:\Program Files\Apache Group\Apache<n>\htdocs
```

where `<n>` is the Apache version number.

For Microsoft Internet Information Services (IIS) this is the `wwwroot` directory which, in a default installation, is located at:

```
C:\Inetpub\wwwroot
```

Starting DDE on Windows

DDE can be started either interactively or as a Windows Service. This section describes starting DDE interactively as part of the installation and testing process. Setting up DDE to start as a Service is described in Appendix B, *DDE as a Windows Service*.

During initial configuration and testing, it is recommended that you start DDE interactively. This will allow you to more easily monitor DDE as it is running and keep track of requests and their outcome. Once it has been established that DDE is running correctly, it can be configured to run as a service.

Reminder: In order to run successfully, DDE must be licensed. Licensing is managed during the ArcIMS Post-Install with the Software Authorization Wizard (SAW). The license file provided to you by ESRI must include the "delivery" extension license; if "delivery" is not shown as being licensed upon completion of the SAW, then DDE will not run.

Interactive Start-up

Here are the steps to start DDE interactively on Windows.

1 First start the web server if it isn't already running.

2 Start DDE from an account that does *not* have Administrator-level privileges. This ensures that the FME Server component is not running as an Administrator.

Warning

This is a safety precaution because the FME Server periodically deletes all files from the results directory when they reach a certain age. If you start DDE as an Administrator, and if erroneous settings were made during installation, the FME Server could possibly delete many other files.

From the Windows Start button, select **Programs | DDE | Start DDE.**

A DOS command window is created, containing various status messages. From this window, you can monitor the start-up messages. After about 30 seconds, DDE's start-up procedure is complete and ready for use.

If there is a problem with the start-up process:

If the startup messages fail to cease scrolling after approximately 30 seconds, it indicates that a problem has occurred that is preventing complete start-up from being achieved.

There are two common causes of this behavior: either the FME Server is improperly licensed, or one or more required TCP/IP ports are already in use and cannot be accessed by DDE.

In either case, the DDE startup sequence will be in an endless restart loop. Cancel the loop by clicking on the Windows Start button, and selecting **Programs | DDE | Stop DDE**. This sends a shutdown request to DDE.

Once DDE start-up is halted, you should first ensure that the FME Server is properly licensed by following the procedure below:

- Using a file browser, navigate to the `fme` directory within the main DDE installation directory.
- Double-click on the `fmegui.exe` file icon, bringing up the FME interactive GUI window. Then click on the window's **Help > About** menu item. The resulting information panel will display license information for the FME Server if the license is installed and valid. If the license information is not listed, then you should obtain a valid FME Server license from your representative, and then install it.

If you have confirmed that the FME Server is properly licensed, then you should investigate possible port usage conflicts. These are usually caused by a previously running instance of DDE that was shut down incompletely, leaving one or more ports still locked by the partially running DDE processes.

First confirm that DDE is not already running as a background Windows Service. This can be determined by running the Windows Service Manager and checking the ArcIMSDDE entry in the services list. If this service is in the running state, then the DDE is already running.

You can either leave it running and use it as-is, or, if desired, you can stop the service and then restart with an interactive start-up using the "Start DDE" menu command.

If DDE is not already running as a service, then you can use the Windows Task Manager to see if any standalone DDE processes are still running. These will show up in the Task Manager process list as one, two or three `java.exe` processes and an `fme.exe` process. You can use the Task Manager to terminate these processes prior to restarting DDE.

Note If any of the DDE components fails to start (for example, if DDE is not licensed) the Process Monitor will keep trying to restart the failed component and you will notice continually scrolling start-up and error messages in the DDE status DOS window. If this occurs, stop DDE (See *Stopping DDE on Windows* on page 42), resolve the problem, and restart DDE.

Logging into DDE on Windows

Logging in to DDE is only required when DDE is being run as a standalone system (prior to integration with ArcIMS). Once DDE configuration and integration with ArcIMS has been completed, all interaction with DDE should be via the ArcIMS web map page.

The following steps will help to confirm successful operation of DDE at this stage of the installation process and should be completed prior to continuing with the configuration of DDE.

1. Start a web browser, such as Netscape or Internet Explorer, and enter the following URL into the browser's address field:

   ```
   http://<hostname>/safeViewerHTML/login.html
   ```

   Replace `<hostname>` with the network hostname of the system on which DDE is being run. For example, if the DDE hostname is `SPOCK,` you would enter:

   ```
   http://SPOCK/safeViewerHTML/login.html
   ```

2. Log in to the initial web page with the `guest` ID or any other valid ID. You can also enter an e-mail address to which DDE can send result notifications. (By default, however, DDE does not send notification e-mail. Please see *Result Notification via E-mail* on page 228, for details.)

3. Once you are logged in, you should see the Index Map page. Click on a grid square in the index map to zoom into that area. If you can successfully zoom into an area, then basic operation of DDE has been confirmed, and you may continue with configuration of DDE for user source data and integration of DDE with ArcIMS.

Stopping DDE on Windows

To stop DDE when you have started it interactively, click on the Windows Start button, and select **Programs | DDE | Stop DDE**.

After a short pause, a new shutdown command window will appear and the main DDE window will display a series of messages and disappear. The shutdown command window will then display a shutdown success message and also disappear. At this point DDE has shut down.

Appendix B, *DDE as a Windows Service* describes how to shut down DDE if it was started as a Windows Service.

Warning

You MUST use the Windows **Stop DDE** command when stopping DDE. If you do not use this command, DDE may fail to stop completely and it will be impossible to restart DDE.

Do NOT attempt to stop DDE by typing Ctrl-C in the DDE status window or by closing the status window.

For details on recovering from an erroneously stopped DDE instance, refer to *Appendix F, Troubleshooting*.

Uninstalling DDE from Windows

To uninstall the DDE components, select:

Start | Programs | DDE | Uninstall DDE

After the uninstallation procedure completes, it will likely report a number of remaining files that must be deleted manually.

Note This procedure uninstalls the Process Monitor, QServer and Translation Servlet components of DDE. It does not uninstall any web server.

Installing DDE on a Single UNIX System

1 For ease of maintenance, DDE should be installed from a user account that has been set up specifically for DDE operations. This account should not be the root account, but it must have write access permission to:
   - the webserver 's public documents root directory
   - the webserver's CGI scripts root directory
   - the translation result subdirectory's root directory.

   These directories are specified during the editing of the configuration script.

   When run, the configuration script first checks to see if the user account has write permission to these directories and if not, terminates with a warning message without performing the installation. If this occurs, please ensure that the account running the configuration script has write permission to the specified directories and then run the script again.

2 Create a DDE installation directory (referred to as `<DDEInstallDir>` in this document).

3 Copy the following zip files from the DDE distribution into `<DDEInstallDir>`:
```
DDEUnixConfig.zip
DDEUnixMain.zip
```

4 cd to `<DDEInstallDir>`.

5 Unzip `DDEUnixConfig.zip` into `<DDEInstallDir>`. For UNIX environments that do not supply the unzip utility, DDE includes four UNIX-specific binaries of the unzip utility on the distribution CD, called `unzip<UnixVariant>`. These can be copied to your DDE installation and used instead, if desired. Please use the appropriate one to unzip the `DDEUnixConfig.zip` file. For example:
```
unzipSolaris DDEUnixConfig.zip
```

6 Issue a `chmod -R u+x config` command. This sets execute access for the configuration files just unzipped.

7 cd to `config`.

8 Edit the `configureDDE` script file by assigning site-specific values to the configuration variables.

Warning

- You **must** supply appropriate values for all of the configuration variables.
- Do not comment out any of these variables.
- Do not leave these variables with undefined, blank or null values.
- Failure to supply actual values for any of these variables can result in MAJOR FILE DELETIONS on your system.
- This can occur when the `ResultsRootDir` and/or the `ResultsDir` variables have no defined value. The FME Server periodically deletes ALL files within the results directory defined by these two variables. If these variables have no defined values, the results directory becomes the "/" root directory and the FME Server attempts to recursively delete all files in the system's root directory.

Configuration variables are listed in the table below, followed by an example configuration setup.

Configuration Variable	Description
`HostName`	The network name of the DDE system. In most cases this is the same system on which the webserver is running.
`WebServerPort`	The port number that the web server listens on for browser requests. For non-secure (http:) environments the usual default value is 80. For secure (https:) environments, the usual default value is 443.
`ServletPort`	The port the translation servlet listens on for translation requests. The usual default value is 8194 for non-secure (http) environments using the servletrunner, and 443 for secure (https) environments where the secure webserver itself runs the servlet.
`QServerHostName`	The network name of the system on which the QServer is running.
`WebHTMLDir`	The full pathname of the webserver's public HTML root directory. For the Apache webserver, `WebHTMLDir` usually refers to the `htdocs` subdirectory.
`ResultsRootDir`	The full pathname of the server's public root directory containing the `ResultsDir` translation result subdirectory (see below). For the Apache webserver, `ResultsRootDir` usually refers to the `htdocs` subdirectory.

Configuration Variable	Description
`ResultsDir`	The name of the translation result subdirectory located within the `ResultsRootDir` (see above). The `ResultsDir` contains the actual translation results.
`SDInstallDir`	The full pathname of the DDE installation directory created to hold the DDE files.
`FMEInstallDir`	The full pathname of the DDE FME server installation directory. The FME is installed automatically as part of the overall DDE installation in a subdirectory called `fme` located within the main DDE installation directory. This FME must be licensed for use with DDE.
`ProtocolPrefixForResults`	The web protocol that is used to access the translation results. Currently allowed values are `http:`, `https:` and `ftp:`. Usually `http:` or `https:` is specified. The trailing colon character is a required part of the prefix value.
`Protocol PrefixForConfigFile`	The web protocol that the Servlet will use to access the `config.csv` file via its `layerListConfURL` property. Currently allowed values are `file:`, `http:`, `https:` and `ftp:`. For DDE on a single system, `file:` should be specified. The trailing colon character is a required part of the prefix value.
`ConfigLocation`	The location of the `config.csv` file. This can be an absolute pathname on the local system if the file is local (`protocol = file:`) or a hostname and directory name if the file is remote (`protocol = http: https: or ftp:`). For DDE on a single system using the `file:` protocol, `<configLocation>` is `<DDEInstallDir>/translationControl/system`.
`PurchasingEnabled`	Flag indicating whether or not translation purchasing is enabled. The value must be `false` for DDE and is already preset to this value in this file.
`SMTPHostName`	The hostname of the system running the SMTP mail protocol which should be used to send translation result notification e-mail. For more information on e-mail notification please refer *Asynchronous Result Return* on page 220.
`SMTPSenderAddress`	The e-mail address on the SMTP host which should be used as the sender of translation result notification e-mail.
`AdminPort`	The TCP/IP port on which to listen for admin requests. The default value of 7500 can be used if it does not conflict with any existing port assignments.

Configuration Variable	Description
`TrustedClientIPs`	A comma-separated list of IP addresses in numeric `n.n.n.n` format of clients trusted to send admin commands. If actual IP addresses are not known at configuration time, the default wildcard value of `0.0.0.0` can be used to indicate that ALL clients are trusted. This value can be changed after installation and configuration if desired.
`AdminPassword`	The password string required of clients when sending admin commands.
`CGIScriptRootDir`	The full pathname of the webserver's CGI script root directory. The filepath must be enclosed in double quotation marks.
`CGIScriptRootDir URLName`	The name of the webserver's CGI script root directory as it appears in a URL. It is often `cgi-bin` or `scripts`.
`CGISDScriptSubDirName`	The name of the CGI subdirectory that is to contain the DDE CGI scripts. This subdirectory will be created automatically if it doesn't already exist.

Here's an example configuration setup for DDE on a single system using the Apache webserver:

```
HostName=NOODLE
WebServerPort=80
ServletPort=8194
QServerHostName=NOODLE
WebHTMLDir="/opt2/local/etc/httpd/htdocs"
ResultsRootDir="/opt2/local/etc/httpd/htdocs"
ResultsDir="results"
SDInstallDir="/opt2/DDETest"
FMEInstallDir="/opt2/DDETest/fme"
ProtocolPrefixForResults=http:
ProtocolPrefixForConfigFile=file:
ConfigLocation="/opt2/DDETest/translationControl/system"
PurchasingEnabled=false
SMTPHostName=ANT
SMTPSenderAddress=abc@xyz.com
AdminPort=7500
TrustedClientIPs=173.643.81.3,412.56.87.98
AdminPassword=qwerty
CGIScriptRootDir="/usr/local/ApacheGroup/Apache2/cgi-bin"
CGIScriptRootDirURLName=cgi-bin
CGISDScriptSubDirName=DDE
```

9 Save the modified `configureDDE` script file.

10 As a root user, run the `configureDDE` script file by typing

```
./configureDDE
```

at the command prompt. This unzips the `DDEUnixMain.zip` file, first renaming potentially customized directories with a timestamp suffix if they exist. It then applies the site-specific values assigned above to various DDE files.

During this installation period, numerous messages will scroll by on the screen as DDE files are modified and copied. If the installation pauses during this period and prompts with this type of message:

```
<filepath...> 755  mode ? (y/n)
```

please enter `y` and press **Return** to accept the prompt and continue with the installation.

Note: If the `configureDDE` script fails for any reason, you can safely re-run the script once the error has been resolved.

11 The DDE FME Server component must now be licensed to run in Server mode. To do this, you must run the *ESRI keymanager* utility to add the license file to your system. For example:

```
keymanager -o addfile -i $AIMSHOME -f delivery -v 9 -r
<your_registration_number> -a <your_license_file>.ecp
```

keymanager can be found in `$AIMSHOME/tools`. If you require assistance using *keymanager* or obtaining or locating your licensing information, please contact your local ESRI representative.

Testing Basic DDE Operation

At this point, installation of the basic DDE is complete and it is configured to work with the sample data provided with the application.

Test basic DDE operation by following these three steps:

- Start DDE (see *Starting DDE on UNIX*).
- Log into DDE (see *Logging into DDE on UNIX*).
- Click on a grid square on the Index Map page to zoom in to that area.

If these three steps are successful, then you are ready to move on to configuring the DDE to read user source data.

Starting DDE on UNIX

Interactive Start-up

To start DDE interactively on a UNIX system:

1 First start the web server if it isn't already running. If the Apache web server was installed it may already be running as an automatically started process. Confirm that Apache is running by using a browser to go the URL http://<*apacheHostName*>. The Apache confirmation web page should be displayed, indicating that Apache is running and responding to browser requests.

2 cd to `<DDEInstallDir>`.

3 DDE should be run from the same user account that was used to install it. This keeps the read and write access permissions that were set up during installation consistent with those that are required at runtime. In particular, this enables DDE to write translation results at runtime into the results directory that was created for this purpose during installation.

Please note that the account used to run DDE should not be the root account, for reasons outlined below.

From a ***non-root user account,*** issue the command

```
./startSD&
```

to start DDE in a background process, and then wait until all DDE components have started. The start-up sequence usually completes within one minute. Start-up is complete when no further messages are logged to the screen. There will be some pauses between messages during this sequence. If error messages about patches are displayed during the start-up attempt on a Solaris system, see *Obtaining the Solaris Patches for Java* on page 52.

Note: Running from a non-root user account ensures that the FME Server component is not running as root. This is a safety precaution because the FME Server periodically deletes all files in the translation results directory when they reach a certain age. If this is done as root, and if erroneous settings were made during installation, the FME Server could possibly delete many other files. Please see the *Warning* on page 44.

Reminder: In order to run successfully, DDE must be licensed. Please refer to Step 11 of "Installing DDE on Unix" for instructions on how to license the DDE.

If there is a problem with the start-up process:

If the start-up messages fail to cease scrolling after approximately 30 seconds, it indicates that a problem has occurred that is preventing complete start-up from being achieved.

There are two common causes of this behavior: either the FME Server is improperly licensed, or one or more required TCP/IP ports are already in use and cannot be accessed by DDE.

In either case the DDE start-up sequence will be in an endless restart loop which should first be cancelled by running the DDE stop procedure. This is done by going to the main DDE installation directory and running the `stopSD` shell script. This runs a program which sends a shutdown request to the DDE.

Once DDE start-up is halted, you should first ensure that the FME Server is properly licensed by following the procedure below:

- From a UNIX command line, change to the `fme` directory within the main DDE installation directory.
- On the command line within the `fme` directory, enter the `runfme` command.

FME will start and after a short pause, will display licensing information. If the information indicates that it is licensed for FME Server operation, then FME licensing is correct. If the license information is not listed, then you should obtain a valid FME Server license from your representative, and then install it.

If you have confirmed that the FME Server is properly licensed, then you should investigate possible port usage conflicts. These are usually caused by a previously running instance of DDE that was shut down incompletely, leaving one or more ports still locked by the partially running DDE processes.

First confirm from your system management that DDE is not already running as an automatic background boot-time job.

If DDE is not already running as a background job, then issue a process display command from the command line to list all DDE-related processes that may still be running. The exact syntax of the command will depend on the particular version of UNIX in use. What is desired is to have the displayed process list the process group ID, or PGID, of each process associated with a command string that includes DDE. Here is an example for Solaris:

```
ps -jal | grep DDE | more
```

Once the PGID of the DDE-related processes has been determined in this way, the "kill" command can be used to terminate all processes in the DDE job tree identified by the single PGID number. The command would typically have this form:

```
kill -9 -<pgid>
```

Automatic Start-up from a Shell Script

Because UNIX site environments can differ widely from each other, there is no one best method for starting DDE automatically from a shell script. However, the following steps outline one possible approach that can be used as a starting point:

1. After DDE has been installed, create a user account to be used exclusively for running DDE. This user account should *not* have root-level privileges. However, it does require write access to the translation results directory, which is typically located within the web server's root directory.
2. Change the ownership of all installed DDE directories and files (including the `results` and `safeViewerHTML` directories created in the web server root) to be owned by the DDE user account.
3. From a shell script, issue the following command to start DDE under the `<SDuser>` account:

   ```
   su - <SDuser> -c <DDEInstallDir>/startSD > <outputLogFile> &
   ```

   This will create a number of processes for the complete default DDE installation. These processes can be listed using the following or similar command, depending on the shell in use:

   ```
   ps -f -u <SDuser>
   ```

Logging into DDE on UNIX

Logging in to DDE is only required when DDE is being run as a standalone system (prior to integration with ArcIMS). Once DDE configuration and integration with ArcIMS has been completed, all interaction with DDE should be via the ArcIMS web map page.

The following steps will help to confirm successful operation of DDE at this stage of the installation process, and should be completed prior to continuing with the configuration of DDE.

1. To log into DDE, start a web browser that has URL access to the DDE system. In the browser, go to URL:

   ```
   http://<hostname>/safeViewerHTML/login.html
   ```

   For example, if DDE was installed on system `GREEN`, the URL to it would be:

   ```
   http://GREEN/safeViewerHTML/login.html
   ```

2. Log in to the initial web page with the `guest` ID or any other ID, such as your first name. Also enter an e-mail address to which DDE can send result notifications. (By default, however, DDE does not send notification e-mail. Please see *Result Notification via E-mail* on page 228, for details.)

3 Once you are logged in, you should see the Index Map page. Click on a grid square in the index map to zoom into that area. If you can successfully zoom into an area, then basic operation of the DDE has been confirmed, and you may continue with configuration of DDE for user source data and integration of DDE with ArcIMS.

Required Java Patches for UNIX

Certain patches are sometimes required on UNIX systems for Java to run correctly. When such patches are absent, attempts to start DDE will display an error message indicating that patches need to be installed.

The required patches can be specific for different combinations of UNIX versions and Java versions.

If you see patch messages when starting DDE on a UNIX system, you need to obtain and install the patches before starting DDE. The sections below describe how to do this for a Solaris system. Other UNIX operating systems are similar.

Determining Your Solaris Version

- The command `uname -a` displays the version of Solaris.
- The version name/number equivalences for Sun's UNIX operating system are:

SunOS 5.3	= Solaris 2.3
SunOS 5.4	= Solaris 2.4
SunOS 5.5	= Solaris 2.5
SunOS 5.5.1	= Solaris 2.5.1
SunOS 5.6	= Solaris 2.6
SunOS 5.7	= Solaris 7

- The command `showrev -p` displays patch and revision information.

Determining Your Solaris DDE JRE/JDK Version

To determine what JRE/JDK version is currently in use with DDE, enter the following UNIX shell command:

```
<DDEInstallDir>/jre/bin/java -version
```

The Java VM will display a version message similar to:

```
java version "1.2.1"
Solaris VM (build Solaris_JDK_1.2.1_04, native threads,
sunwjit)
```

Obtaining the Solaris Patches for Java

To obtain the required patches, you need to go to Sun's web site at:

```
http://www.sun.com/software/solaris/java/download.html
```

1 Find the Java version in use with DDE and select the Solaris SPARC platform edition for the language you want.

   This takes you to a download registration page.

2 Sign up for the free registration and "log on" as if you're going to download the Solaris JDK.

   This takes you to a download page for the Java version used with DDE. From here, you can download a number of things, including the required patches. There are different patch choices depending on the version of Solaris you're running (2.5.1, 2.6, 7, etc.).

3 Download the patch file associated with your particular Solaris version and install the patches it contains according to Sun's included `README` text files.

4 Restart DDE.

Stopping DDE on UNIX

To stop DDE on UNIX, log into the account under which DDE is running, change directory to `<DDEInstallDir>` and issue the following command:

```
./stopSD
```

After a short pause, the command will return with a shutdown success message and DDE will be shut down.

Uninstalling DDE from a UNIX System

To uninstall DDE from a UNIX system, perform following steps:

1 Ensure DDE is shut down. See *Stopping DDE on UNIX* on page 52.

2 Delete the entire DDE directory tree in which DDE is installed.

3 Delete the following directory trees:

```
<webserverDocDir>/safeViewerHTML
<webserverDocDir>/<translationResultsDir>
<webserverCGIDir>/<DDECGIScriptDir>
```

The placeholders used in the above file paths have the following meanings:

- *<webserverDocDir>* refers to the name of the root directory in which the web server stores its publicly accessible document files.
  For example, the Apache web server uses a directory called "htdocs" for this purpose so *<webserverDocDir>* for an Apache web server might look like:
  ```
  /usr/local/apache/htdocs
  ```
- *<translationResultsDir>* refers to the name of the subdirectory in which translation result files are stored. The name would have been user-specified and created automatically during DDE installation, and is typically `translationResults`, `results`, or similar.
- *<webserverCGIDir>* refers to the name of the root directory in which the web server stores its CGI scripts.
  For example, the Apache web server uses a directory called `cgi-bin` for this purpose, so `<webserverCGIDir>` for an Apache web server might look like:
  ```
  /usr/local/apache/cgi-bin
  ```
- *<DDECGIScriptDir>* refers to the name of the subdirectory in which DDE-specific CGI script files are stored. The name would have been user-specified and created automatically during DDE installation, and is typically `DDE`.

4 Disable the DDE export function in ArcIMS by modifying the following ArcIMS file:

```
<ArcIMSMapDir>/ArcIMSparam.js
```

In this file, locate the following line:

```
var useFMEExport=true;
```

and change the "`true`" value to "`false`" so the line now looks like this:

```
var useFMEExport=false;
```

Part 2: Configuring DDE for ESRI ArcIMS

CHAPTER 4

Introduction to DDE for ESRI ArcIMS

The DDE ArcIMS extension allows ArcIMS to communicate with an FME server so that ArcIMS map data may be translated into any of the DDE-supported formats and delivered to the user via the Web.

DDE for ArcIMS makes it possible for an organization to have one central GIS data repository that is served up through ArcIMS to all of its clients over an intranet or the Internet. In order to achieve this functionality DDE for ArcIMS has been developed as a fusion of its two namesake products:

- An ArcIMS website is extended through the addition of JavaScript and HTML components which add functionality to the core product.
- DDE has been modified from its standard distribution so that communication with the `translationServlet` is initiated through the FMEDownload web page, rather than the standard Index Map and Order Form pages.

Supported Configurations

DDE for ArcIMS is supported on all platforms on which ArcIMS 9.0 is supported.

Differences Between Windows and UNIX

Although this document focuses on and provides examples for a Windows installation, the differences between installation for the supported operating systems are minimal. The major difference is the root installation paths for the DDE and ArcIMS products.

Where there is a difference in an installation procedure, it has been noted at the appropriate section of this document.

Note UNIX installation file permissions need to be adequate for both systems to cooperate. It may be worthwhile to create a group where both the ArcIMS and the DDE users (i.e., the user accounts under which these two applications are run) can be members, and enable group file permissions based on this group.

CHAPTER 5

Configuring DDE to Read User Source Data

This chapter gives step-by-step instructions for configuring DDE to read a user source dataset.

Configuration Steps

The main steps involved in configuration are:

1. Prepare for configuration.
2. Define the source data themes.
3. Specify the query coordinate system.
4. Specify the source data spatial extent.
5. Specify the output formats to be made available.
6. Specify the output coordinate systems to be made available.
7. Specify source data format.
8. Restart DDE.

The following sections describe each of these steps.

Prepare for Configuration

As distributed, DDE is initially configured to access a supplied sample source dataset. Before configuring DDE to read a different source dataset, perform the following steps:

- Ensure that DDE has been successfully installed and run with the sample source dataset provided.
- Shut down DDE before configuring for a new source dataset.
- Ensure that the source dataset is correctly set up, and that its spatial extent and coordinate system are known.

Configuring Source Data with the Configuration Tool

Note: This is the recommended method for configuring source data. However, you can also configure source data manually by following the steps in *Configuring Source Data Manually* on page 113. Manual configuration allows DDE to be configured to read from more than one source dataset.

Windows DDE Configuration

This section lists the steps required in Windows to configure DDE for use with user source data.

Source data is restricted to a single format. The following table lists the available formats:

ReaderType	Data Format
`SHAPE`	ESRI Shape
`SDE30`	ESRI Spatial Database Engine 3.x/ArcSDE 8.x
`GEODATABASE_SDE`	ESRI GeoDatabase (SDE)
`GEODATABASE_MDB`	ESRI GeoDatabase (Access) (Windows only)

There are two main steps for configuring the `ReaderType`:

- using the configuration tool `DDEConfig` to input the site-specific configuration parameters
- using Spatial Assistant to define the source data themes

The following pages give the configuration steps for each `ReaderType`.

Configuring ReaderType SHAPE

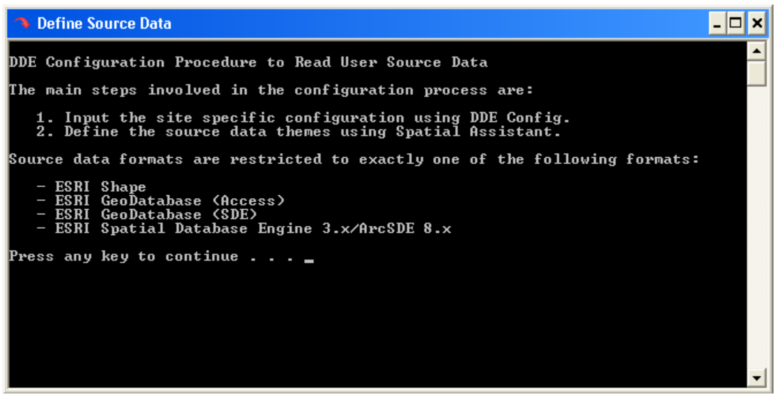
Entering Site-Specific Configuration Parameters:

1 Start the DDE configuration process window `(DDEConfig.bat)` by clicking **Start | Program Files | DDE | Define Source Data**. Alternatively, you can run the tool by locating and double-clicking on this file:

```
<DDEInstallDir>\config\sourceDefiner\DDEConfig.bat
```

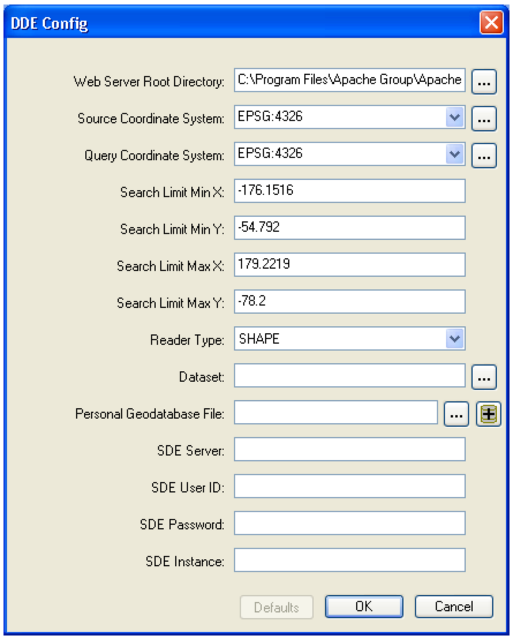
2 Type **Y** to continue the configuration process, or **N** to quit. The DDEConfig.bat process window appears.

Note: This window will remain on your screen as you continue with the configuration steps.



```
DDE Configuration Procedure to Read User Source Data

The main steps involved in the configuration process are:

   1. Input the site specific configuration using DDE Config.
   2. Define the source data themes using Spatial Assistant.

Source data formats are restricted to exactly one of the following formats:

   - ESRI Shape
   - ESRI GeoDatabase (Access)
   - ESRI GeoDatabase (SDE)
   - ESRI Spatial Database Engine 3.x/ArcSDE 8.x

Press any key to continue . . . _
```

3 The **DDE Config** parameter dialog (below) is launched after you press any key in the window shown above. Enter the site-specific configuration parameters in the window and click **OK**.



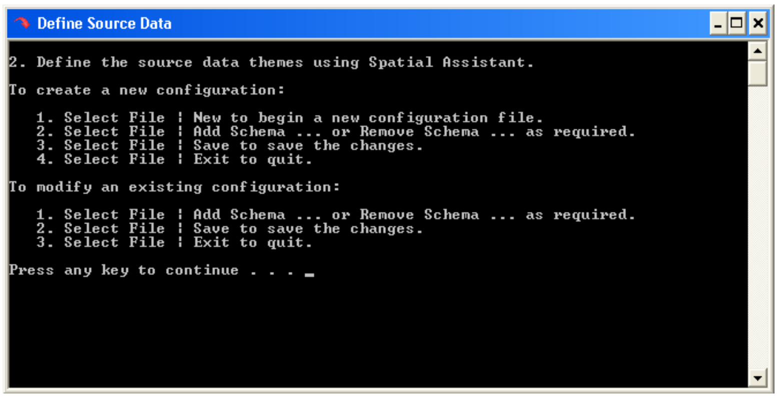
Web Server Root Directory:	C:\Program Files\Apache Group\Apache
Source Coordinate System:	EPSG:4326
Query Coordinate System:	EPSG:4326
Search Limit Min X:	-176.1516
Search Limit Min Y:	-54.792
Search Limit Max X:	179.2219
Search Limit Max Y:	-78.2
Reader Type:	SHAPE
Dataset:	
Personal Geodatabase File:	
SDE Server:	
SDE User ID:	
SDE Password:	
SDE Instance:	

Note that only *certain* site-specific parameters are required for the `SHAPE ReaderType`. These are listed below. All other fields in the dialog are not required for this `ReaderType`, and should remain blank.

- Web Server Root Directory
- Source Coordinate System
- Query Coordinate System
- Search Limit Min X
- Search Limit Min Y
- Search Limit Max X
- Search Limit Max Y
- ReaderType
- Dataset (the directory containing the Shapefile data source)

Defining the Source Data Themes:

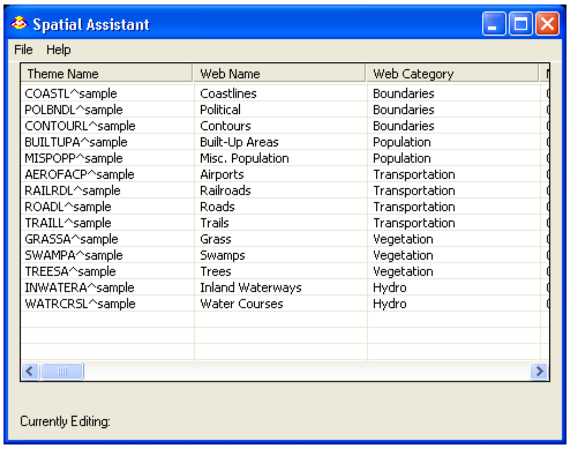
4 After the DDE Config parameter dialog has been dismissed, the `DDEConfig` process displays the following screen, describing how to use Spatial Assistant to define source data themes. When you press a key to continue, the `DDEConfig` process will launch the Spatial Assistant application.



Define Source Data

```
2. Define the source data themes using Spatial Assistant.

To create a new configuration:

   1. Select File | New to begin a new configuration file.
   2. Select File | Add Schema ... or Remove Schema ... as required.
   3. Select File | Save to save the changes.
   4. Select File | Exit to quit.

To modify an existing configuration:

   1. Select File | Add Schema ... or Remove Schema ... as required.
   2. Select File | Save to save the changes.
   3. Select File | Exit to quit.

Press any key to continue . . . _
```

Initially Spatial Assistant will display either the sample data `config.csv` contents or the `config.csv` contents from the most recently saved changes.



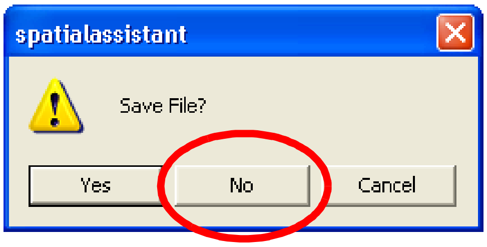
Spatial Assistant

File Help

Theme Name	Web Name	Web Category
COASTL^sample	Coastlines	Boundaries
POLBNDL^sample	Political	Boundaries
CONTOURL^sample	Contours	Boundaries
BUILTUPA^sample	Built-Up Areas	Population
MISPOPP^sample	Misc. Population	Population
AEROFACP^sample	Airports	Transportation
RAILRDL^sample	Railroads	Transportation
ROADL^sample	Roads	Transportation
TRAILL^sample	Trails	Transportation
GRASSA^sample	Grass	Vegetation
SWAMPA^sample	Swamps	Vegetation
TREESA^sample	Trees	Vegetation
INWATERA^sample	Inland Waterways	Hydro
WATRCRSL^sample	Water Courses	Hydro

Currently Editing:

- To create a new source data configuration, select **File | New** from Spatial Assistant. This creates a new `config.csv` file. Click **No** when prompted to save the previous `config.csv` file:

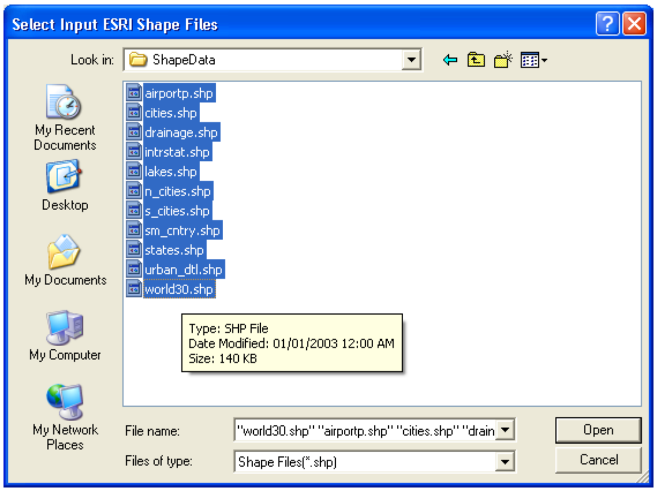


spatialassistant

Save File?

5 Add themes to `config.csv` by selecting **File | New** or **File | Add Schema** from Spatial Assistant. You will be prompted to specify an Input Data Source.

- Click the Browse button [...] in the Dataset field to display the following file chooser.



airportp.shp
cities.shp
drainage.shp
intrstat.shp
lakes.shp
n_cities.shp
s_cities.shp
sm_cntry.shp
states.shp
urban_dtl.shp
world30.shp

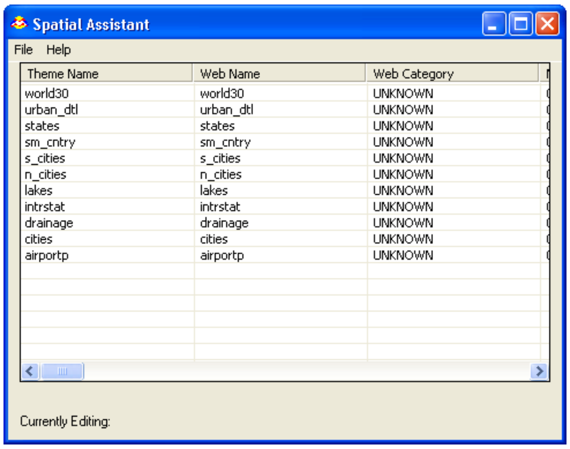
Type: SHP File
Date Modified: 01/01/2003 12:00 AM
Size: 140 KB

File name: "world30.shp" "airportp.shp" "cities.shp" "drain

Files of type: Shape Files(*.shp)

- Select the desired `*.shp` files and click **Open**.
- Click **OK** to confirm your selection.

The selected themes are updated in the `config.csv` file as shown in Spatial Assistant:



Theme Name	Web Name	Web Category
world30	world30	UNKNOWN
urban_dtl	urban_dtl	UNKNOWN
states	states	UNKNOWN
sm_cntry	sm_cntry	UNKNOWN
s_cities	s_cities	UNKNOWN
n_cities	n_cities	UNKNOWN
lakes	lakes	UNKNOWN
intrstat	intrstat	UNKNOWN
drainage	drainage	UNKNOWN
cities	cities	UNKNOWN
airportp	airportp	UNKNOWN

Currently Editing:

6 Exit Spatial Assistant by closing the window, or through the **File** menu. Select **Yes** when prompted to save the file.

The `DDEConfig.bat` screen will display:

```
Continue the configuration process (Y/N)?
Y to continue, N to exit.
```

7 Type **Y** and press the Enter key. The `DDEConfig.bat` window will display:

```
Configuration Complete.
Done.
```

DDE has now been configured to recognize your user source data but further configuration steps are required in order to integrate DDE with ArcIMS.

To complete the integration process, please continue to *Integrating DDE with ArcIMS* on page 81.

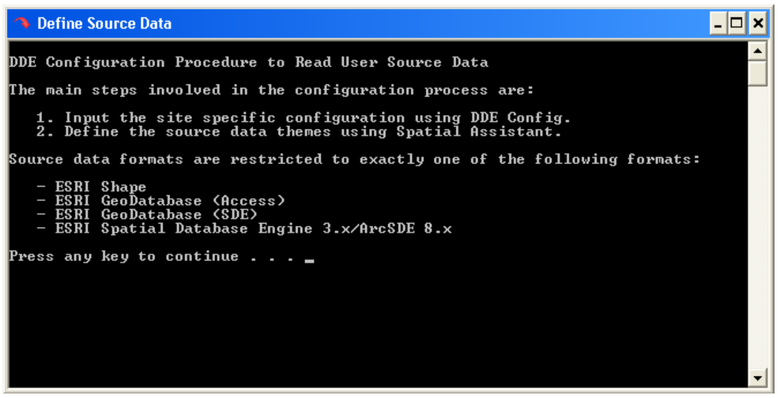
Configuring ReaderType SDE30

Entering Site-Specific Configuration Parameters:

1 Start the DDE configuration process window by clicking **Start | Program Files | DDE | Define Source Data**. Alternatively, you can run the tool by locating and double-clicking on this file:

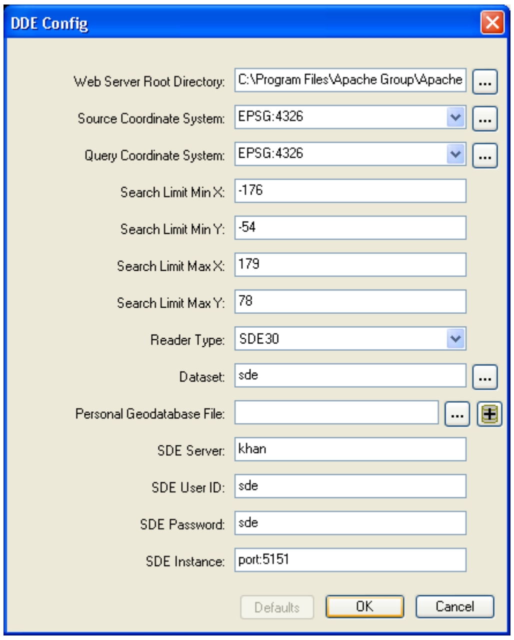
```
<DDEInstallDir>\config\sourceDefiner\DDEConfig.bat
```

2 Type **Y** to continue the configuration process, or **N** to quit. The DDEConfig.bat window appears. **Note:** This window will remain on your screen as you continue with the configuration steps.



```
DDE Configuration Procedure to Read User Source Data

The main steps involved in the configuration process are:

   1. Input the site specific configuration using DDE Config.
   2. Define the source data themes using Spatial Assistant.

Source data formats are restricted to exactly one of the following formats:

   - ESRI Shape
   - ESRI GeoDatabase (Access)
   - ESRI GeoDatabase (SDE)
   - ESRI Spatial Database Engine 3.x/ArcSDE 8.x

Press any key to continue . . . _
```

3 The **DDE Config** parameter dialog (below) is launched after you press any key in the window shown above. Enter the site-specific configuration parameters in the window and click **OK**.



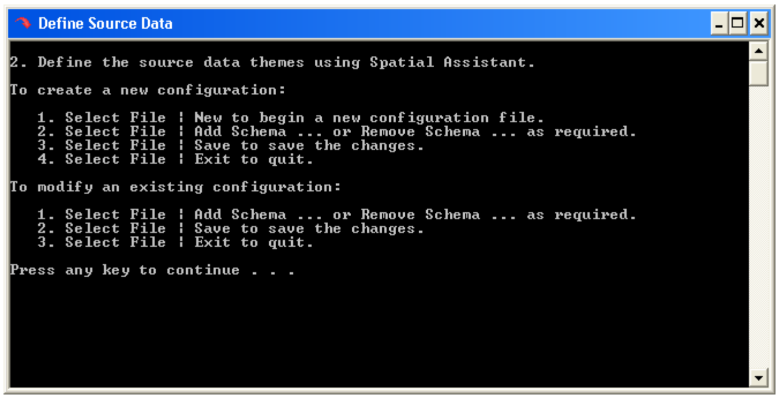
Web Server Root Directory:	C:\Program Files\Apache Group\Apache
Source Coordinate System:	EPSG:4326
Query Coordinate System:	EPSG:4326
Search Limit Min X:	-176
Search Limit Min Y:	-54
Search Limit Max X:	179
Search Limit Max Y:	78
Reader Type:	SDE30
Dataset:	sde
Personal Geodatabase File:	
SDE Server:	khan
SDE User ID:	sde
SDE Password:	sde
SDE Instance:	port:5151

Note that only *certain* site-specific parameters are required for the `SDE30 ReaderType`. These are listed below. All other fields in the dialog are not required for this `ReaderType`, and should remain blank.

- Web Server Root Directory
- Source Coordinate System
- Query Coordinate System
- Search Limit Min X
- Search Limit Min Y
- Search Limit Max X
- Search Limit Max Y
- Reader Type
- Dataset (the name of the database dataset; e.g., *`DDE`*)
- SDE Server
- SDE User ID
- SDE Password
- SDE Instance

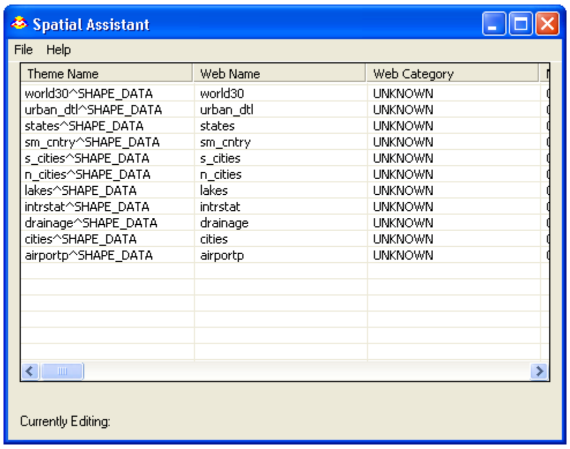
Defining the Source Data Themes:

4 After the DDE Config parameter dialog has been dismissed, the `DDEConfig` process displays the following screen, describing how to use Spatial Assistant to define source data themes. When you press a key to continue, the `DDEConfig` process will launch the Spatial Assistant application.



```
2. Define the source data themes using Spatial Assistant.
To create a new configuration:

   1. Select File | New to begin a new configuration file.
   2. Select File | Add Schema ... or Remove Schema ... as required.
   3. Select File | Save to save the changes.
   4. Select File | Exit to quit.

To modify an existing configuration:

   1. Select File | Add Schema ... or Remove Schema ... as required.
   2. Select File | Save to save the changes.
   3. Select File | Exit to quit.

Press any key to continue . . .
```

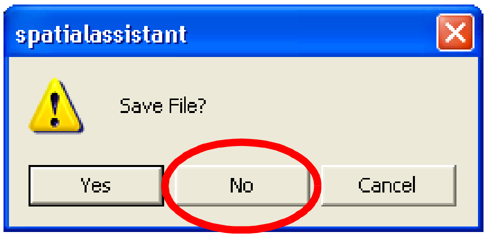
Initially Spatial Assistant will display either the sample data `config.csv` contents or the `config.csv` contents from the most recently saved changes..



Theme Name	Web Name	Web Category
world30^SHAPE_DATA	world30	UNKNOWN
urban_dtl^SHAPE_DATA	urban_dtl	UNKNOWN
states^SHAPE_DATA	states	UNKNOWN
sm_cntry^SHAPE_DATA	sm_cntry	UNKNOWN
s_cities^SHAPE_DATA	s_cities	UNKNOWN
n_cities^SHAPE_DATA	n_cities	UNKNOWN
lakes^SHAPE_DATA	lakes	UNKNOWN
intrstat^SHAPE_DATA	intrstat	UNKNOWN
drainage^SHAPE_DATA	drainage	UNKNOWN
cities^SHAPE_DATA	cities	UNKNOWN
airportp^SHAPE_DATA	airportp	UNKNOWN

Currently Editing:

- To create a new source data configuration, select **File | New** from Spatial Assistant. This creates a new `config.csv` file. Click **No** when prompted to save the previous `config.csv` file:



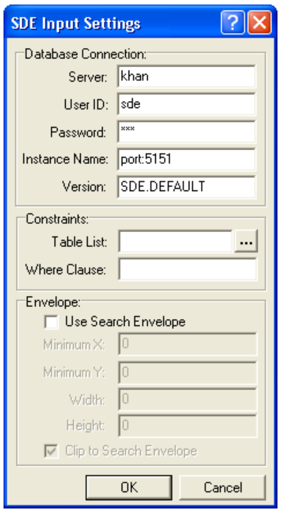
spatialassistant

Save File?

5 Add themes to `config.csv` by selecting **File | New** or **File | Add Schema** from Spatial Assistant.

You will be prompted to specify an Input Data Source.

- Click the **Settings** button in the Input Data Source dialog to locate the database file. The SDE Input Settings dialog appears.



SDE Input Settings

Database Connection:

Server:	khan
User ID:	sde
Password:	***
Instance Name:	port:5151
Version:	SDE.DEFAULT

Constraints:

Table List:	
Where Clause:	

Envelope:

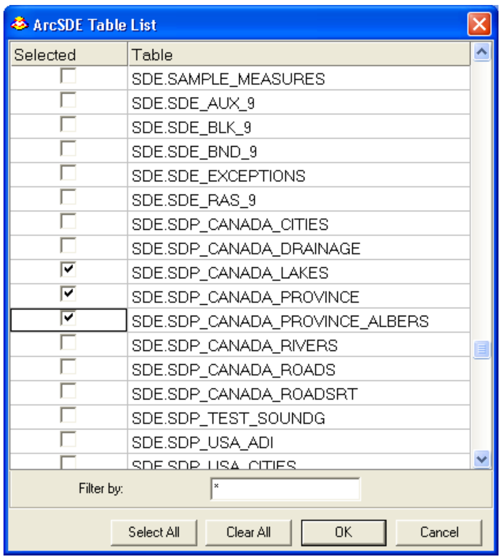
☐ Use Search Envelope

Minimum X:	0
Minimum Y:	0
Width:	0
Height:	0

☑ Clip to Search Envelope

- Click the Table List Browse button [...] to extract the desired SDE30 tables.

- The ArcSDE table list is extracted. Check the desired SDE30 layers. Click **OK** to close all open dialogs.



ArcSDE Table List

Selected	Table
☐	SDE.SAMPLE_MEASURES
☐	SDE.SDE_AUX_9
☐	SDE.SDE_BLK_9
☐	SDE.SDE_BND_9
☐	SDE.SDE_EXCEPTIONS
☐	SDE.SDE_RAS_9
☐	SDE.SDP_CANADA_CITIES
☐	SDE.SDP_CANADA_DRAINAGE
✓	SDE.SDP_CANADA_LAKES
✓	SDE.SDP_CANADA_PROVINCE
✓	SDE.SDP_CANADA_PROVINCE_ALBERS
☐	SDE.SDP_CANADA_RIVERS
☐	SDE.SDP_CANADA_ROADS
☐	SDE.SDP_CANADA_ROADSRT
☐	SDE.SDP_TEST_SOUNDG
☐	SDE.SDP_USA_ADI
☐	SDE.SDP_USA_CITIES

Filter by: *

6 Exit Spatial Assistant by closing the window, or through the **File** menu. Select **Yes** when prompted to save the file.

The selected themes are updated in `config.csv`.

The `DDEConfig.bat` screen will display:

```
Continue the configuration process (Y/N)?
Y to continue, N to exit.
```

7 Type **Y**. The `DDEConfig.bat` window will display:

```
Configuration Complete.
Done.
```

DDE has now been configured to recognize your user source data but further configuration steps are required in order to integrate DDE with ArcIMS.

To complete the integration process, please continue to *Integrating DDE with ArcIMS* on page 81.

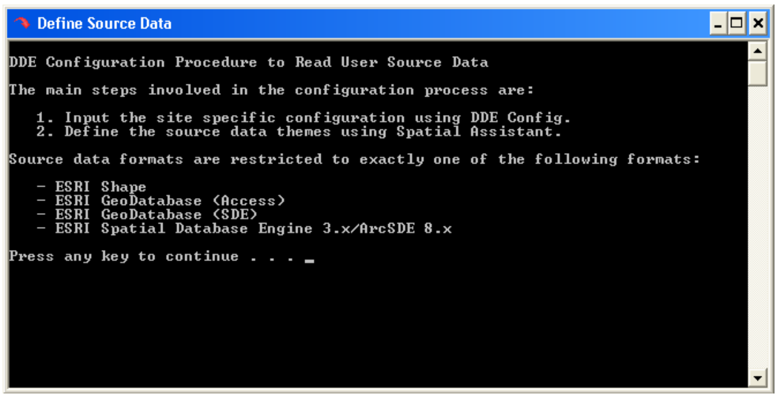
Configuring ReaderType GEODB_MDB

Entering Site-Specific Configuration Parameters:

1 Start the DDE configuration process window by clicking **Start | Program Files | DDE | Define Source Data**. Alternatively, you can run the tool by locating and double-clicking on this file:

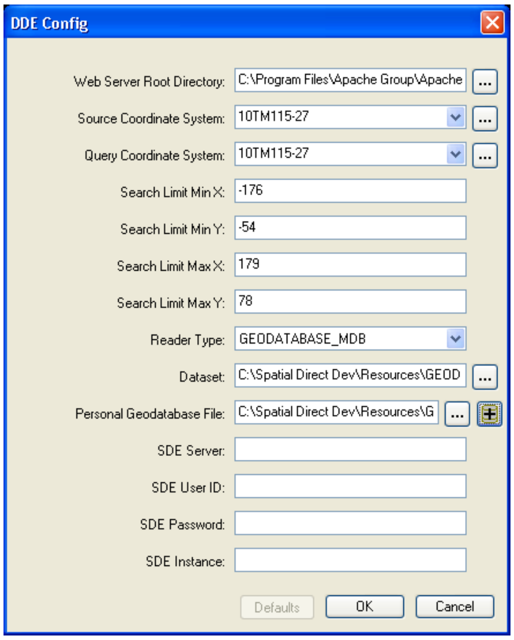
```
<DDEInstallDir>\config\sourceDefiner\DDEConfig.bat
```

2 Type **Y** to continue the configuration process, or **N** to quit. The DDEConfig.bat window appears. **Note:** This window will remain on your screen as you continue with the configuration steps.



```
DDE Configuration Procedure to Read User Source Data
The main steps involved in the configuration process are:

   1. Input the site specific configuration using DDE Config.
   2. Define the source data themes using Spatial Assistant.

Source data formats are restricted to exactly one of the following formats:

   - ESRI Shape
   - ESRI GeoDatabase (Access)
   - ESRI GeoDatabase (SDE)
   - ESRI Spatial Database Engine 3.x/ArcSDE 8.x

Press any key to continue . . .
```

3 The **DDE Config** parameter dialog (below) is launched after you press any key in the window shown above. Enter the site-specific configuration parameters in the window and click **OK**.



Web Server Root Directory:	C:\Program Files\Apache Group\Apache
Source Coordinate System:	10TM115-27
Query Coordinate System:	10TM115-27
Search Limit Min X:	-176
Search Limit Min Y:	-54
Search Limit Max X:	179
Search Limit Max Y:	78
Reader Type:	GEODATABASE_MDB
Dataset:	C:\Spatial Direct Dev\Resources\GEOD
Personal Geodatabase File:	C:\Spatial Direct Dev\Resources\G
SDE Server:	
SDE User ID:	
SDE Password:	
SDE Instance:	

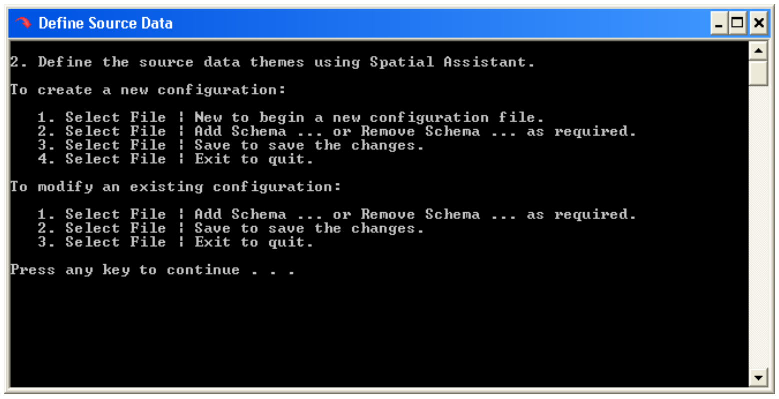
Note that only *certain* site-specific parameters are required for the `GEODB_MDB ReaderType`. These are listed below. All other fields in the dialog are not required for this `ReaderType`, and should remain blank.

- Web Server Root Directory
- Source Coordinate System

- Query Coordinate System
- Search Limit Min X
- Search Limit Min Y
- Search Limit Max X
- Search Limit Max Y
- Reader Type
- Dataset (the directory containing the `.mdb` file)
- Personal Geodatbase File (`.mdb` file selection with the Dataset defined directory)

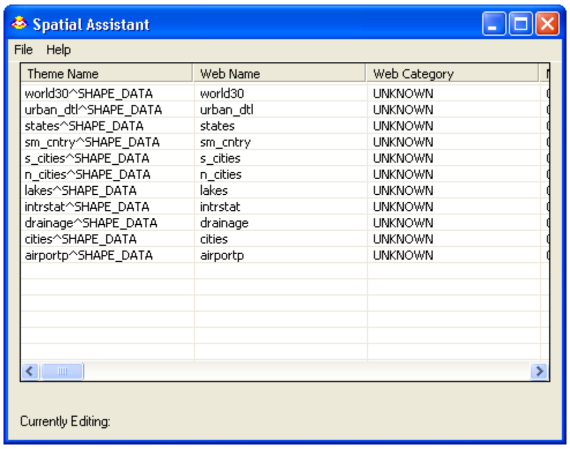
Defining the Source Data Themes:

4 After the DDE Config parameter dialog has been dismissed, the `DDEConfig` process displays the following screen, describing how to use Spatial Assistant to define source data themes. When you press a key to continue, the `DDEConfig` process will launch the Spatial Assistant application.



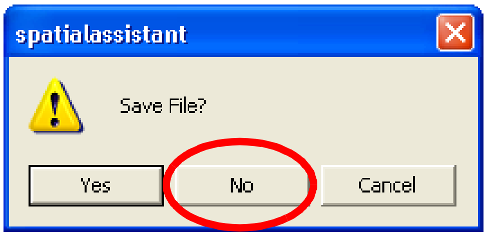
```
2. Define the source data themes using Spatial Assistant.

To create a new configuration:

   1. Select File | New to begin a new configuration file.
   2. Select File | Add Schema ... or Remove Schema ... as required.
   3. Select File | Save to save the changes.
   4. Select File | Exit to quit.

To modify an existing configuration:

   1. Select File | Add Schema ... or Remove Schema ... as required.
   2. Select File | Save to save the changes.
   3. Select File | Exit to quit.

Press any key to continue . . .
```

Initially Spatial Assistant will display either the sample data `config.csv` contents or the `config.csv` contents from the most recently saved changes.



Theme Name	Web Name	Web Category
world30^SHAPE_DATA	world30	UNKNOWN
urban_dtl^SHAPE_DATA	urban_dtl	UNKNOWN
states^SHAPE_DATA	states	UNKNOWN
sm_cntry^SHAPE_DATA	sm_cntry	UNKNOWN
s_cities^SHAPE_DATA	s_cities	UNKNOWN
n_cities^SHAPE_DATA	n_cities	UNKNOWN
lakes^SHAPE_DATA	lakes	UNKNOWN
intrstat^SHAPE_DATA	intrstat	UNKNOWN
drainage^SHAPE_DATA	drainage	UNKNOWN
cities^SHAPE_DATA	cities	UNKNOWN
airportp^SHAPE_DATA	airportp	UNKNOWN

- To create a new source data configuration, select **File | New** from Spatial Assistant. This creates a new `config.csv` file. Click **No** when prompted to save the previous `config.csv` file:



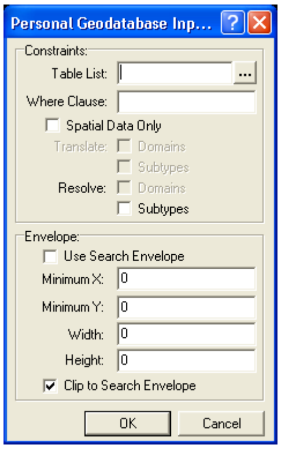
Save File?

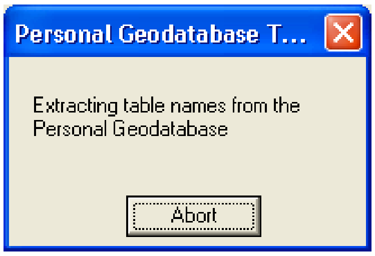
5 Add themes to `config.csv` by selecting **File | New** or **File | Add Schema** from Spatial Assistant.

You will be prompted to specify an Input Data Source.

6 Click the **Settings** button in the Input Data Source dialog to locate the **.mdb** database file. The Input Settings dialog appears.

- Click the Table List Browse button ... to extract the desired data tables.





Extracting table names from the Personal Geodatabase

The Personal Geodatabase table list is extracted.

7 Check the desired tables to add.

8 Click **OK** to close all open dialogs.

9 Exit Spatial Assistant by closing the window, or through the **File** menu. Select **Yes** when prompted to save the file.

The selected themes are updated in `config.csv`.

The `DDEConfig.bat` screen will display:

```
Continue the configuration process (Y/N)?
Y to continue, N to exit.
```

10 Type **Y**. The `DDEConfig.bat` window will display:

```
Configuration Complete.
Done.
```

DDE has now been configured to recognize your user source data but further configuration steps are required in order to integrate DDE with ArcIMS.

To complete the integration process, please continue to *Integrating DDE with ArcIMS* on page 81.

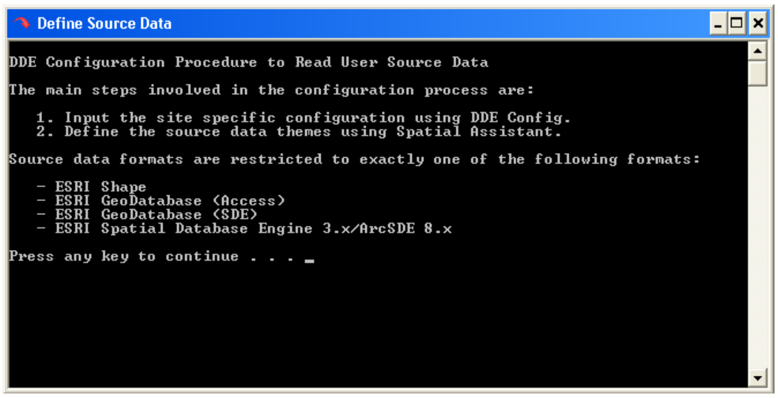
Configuring ReaderType GEODATABASE_SDE

Entering Site-Specific Configuration Parameters:

1 Start the DDE configuration process window by clicking **Start | Program Files | DDE | Define Source Data**. Alternatively, you can run the tool by locating and double-clicking on this file:

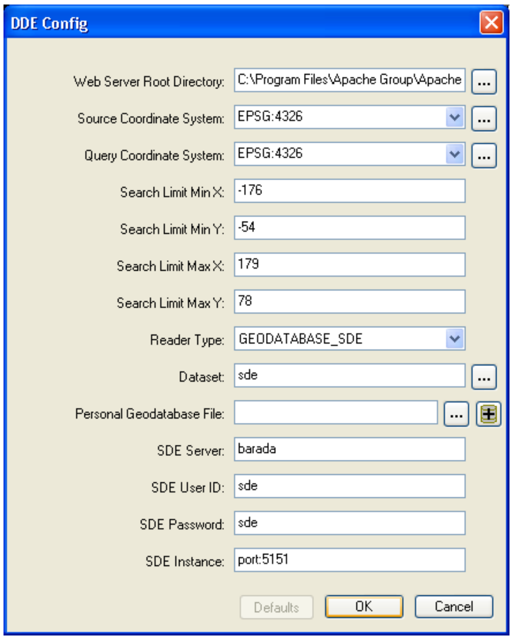
```
<DDEInstallDir>\config\sourceDefiner\DDEConfig.bat
```

2 Type **Y** to continue the configuration process, or **N** to quit. The DDEConfig.bat window appears. **Note:** This window will remain on your screen as you continue with the configuration steps.



```
DDE Configuration Procedure to Read User Source Data

The main steps involved in the configuration process are:

   1. Input the site specific configuration using DDE Config.
   2. Define the source data themes using Spatial Assistant.

Source data formats are restricted to exactly one of the following formats:

   - ESRI Shape
   - ESRI GeoDatabase (Access)
   - ESRI GeoDatabase (SDE)
   - ESRI Spatial Database Engine 3.x/ArcSDE 8.x

Press any key to continue . . . _
```

3 The **DDE Config** parameter dialog (below) is launched after you press any key in the window shown above. Enter the site-specific configuration parameters in the window and click **OK**.



Web Server Root Directory:	C:\Program Files\Apache Group\Apache
Source Coordinate System:	EPSG:4326
Query Coordinate System:	EPSG:4326
Search Limit Min X:	-176
Search Limit Min Y:	-54
Search Limit Max X:	179
Search Limit Max Y:	78
Reader Type:	GEODATABASE_SDE
Dataset:	sde
Personal Geodatabase File:	
SDE Server:	barada
SDE User ID:	sde
SDE Password:	sde
SDE Instance:	port:5151

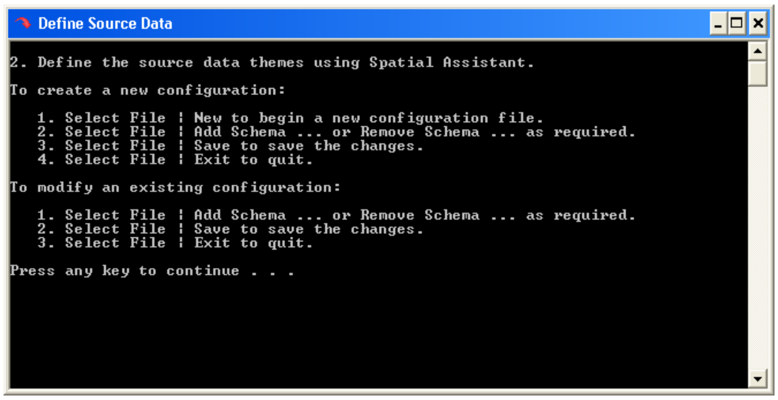
Note that only *certain* site-specific parameters are required for the `GEODATABASE_SDE ReaderType`. These are listed below. All other fields in the dialog are not required for this `ReaderType`, and should remain blank.

- Web Server Root Directory

- Source Coordinate System
- Query Coordinate System
- Search Limit Min X
- Search Limit Min Y
- Search Limit Max X
- Search Limit Max Y
- Reader Type
- Dataset (the name of the database dataset, e.g., `DDE`)
- SDE Server
- SDE User ID
- SDE Password
- SDE Instance

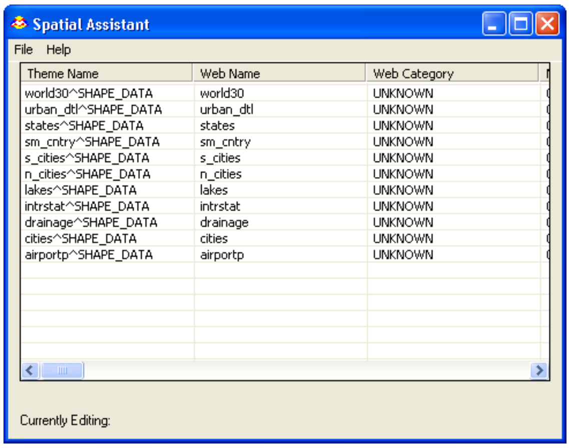
Defining the Source Data Themes:

4 After the DDE Config parameter dialog has been dismissed, the `DDEConfig` process window displays the following screen, describing how to use Spatial Assistant to define source data themes. When you press a key to continue, the `DDEConfig` process will launch the Spatial Assistant application.



```
2. Define the source data themes using Spatial Assistant.

To create a new configuration:

   1. Select File | New to begin a new configuration file.
   2. Select File | Add Schema ... or Remove Schema ... as required.
   3. Select File | Save to save the changes.
   4. Select File | Exit to quit.

To modify an existing configuration:

   1. Select File | Add Schema ... or Remove Schema ... as required.
   2. Select File | Save to save the changes.
   3. Select File | Exit to quit.

Press any key to continue . . .
```

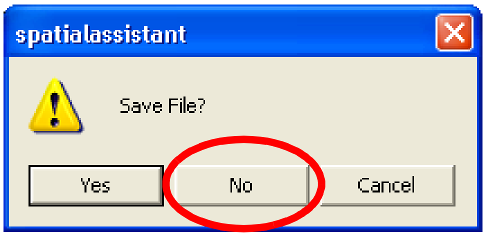
Initially Spatial Assistant will display either the sample data `config.csv` contents or the `config.csv` contents from the most recently saved changes.



Theme Name	Web Name	Web Category
world30^SHAPE_DATA	world30	UNKNOWN
urban_dtl^SHAPE_DATA	urban_dtl	UNKNOWN
states^SHAPE_DATA	states	UNKNOWN
sm_cntry^SHAPE_DATA	sm_cntry	UNKNOWN
s_cities^SHAPE_DATA	s_cities	UNKNOWN
n_cities^SHAPE_DATA	n_cities	UNKNOWN
lakes^SHAPE_DATA	lakes	UNKNOWN
intrstat^SHAPE_DATA	intrstat	UNKNOWN
drainage^SHAPE_DATA	drainage	UNKNOWN
cities^SHAPE_DATA	cities	UNKNOWN
airportp^SHAPE_DATA	airportp	UNKNOWN

Currently Editing:

- To create a new source data configuration, select **File | New** from Spatial Assistant. This creates a new `config.csv` file. Click **No** when prompted to save the previous `config.csv` file:



Save File?

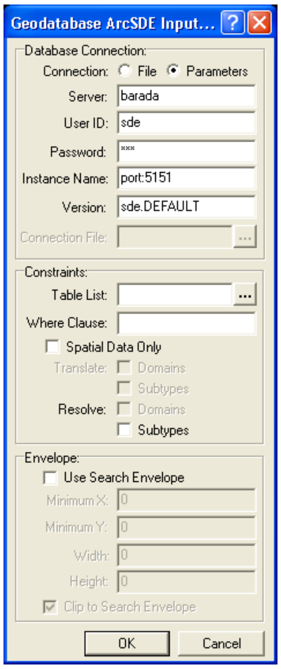
Yes [✓ No] Cancel

5 Add themes to `config.csv` by selecting **File | New** or **File | Add Schema** from Spatial Assistant.

You will be prompted to specify an Input Data Source.

6 Click the **Settings** button in the Input Data Source dialog to locate the **SDE30** data. The Input Settings dialog appears.

- Click the Table List Browse button ... to extract the desired data tables.



The ArcSDE table list is extracted.

7 Check the desired SDE30 layers to add.

8 Click **OK** to close all open dialogs.

9 Exit Spatial Assistant by closing the window, or through the **File** menu. Select **Yes** when prompted to save the file.

The selected themes are updated in `config.csv`.

The `DDEConfig.bat` screen will display:

```
Continue the configuration process (Y/N)?
Y to continue, N to exit.
```

10 Type **Y**. The `DDEConfig.bat` window will display:

```
Configuration Complete.
Done.
```

DDE has now been configured to recognize your user source data but further configuration steps are required in order to integrate DDE with ArcIMS.

To complete the integration process, please continue to *Integrating DDE with ArcIMS* on page 81.

UNIX DDE Configuration

This section lists the steps required in UNIX to configure DDE for use with user source data.

Source data formats are restricted to a single source data format, as listed in the following table:

ReaderType	Data Format
`SHAPE`	ESRI Shape
`SDE30`	ESRI Spatial Database Engine 3.x/ArcSDE 8.x

The following pages give the configuration steps for each `ReaderType`.

1 **Create config.csv.**

Prior to running the DDEConfig.sh shell script the config.csv file must be created as described in *Creation and Modification of config.csv* on page 108.

A utility application called Spatial Assistant is used to create `config.csv`. However, this application must be installed and run on a Windows system, not UNIX. The DDE UNIX distribution includes a Spatial Assistant installer in the form of a subdirectory called `spatialAssistant` on the CD. The contents of this directory should be copied to a Windows system, then Spatial Assistant is installed on that system by double-clicking the `Setup.exe` file.

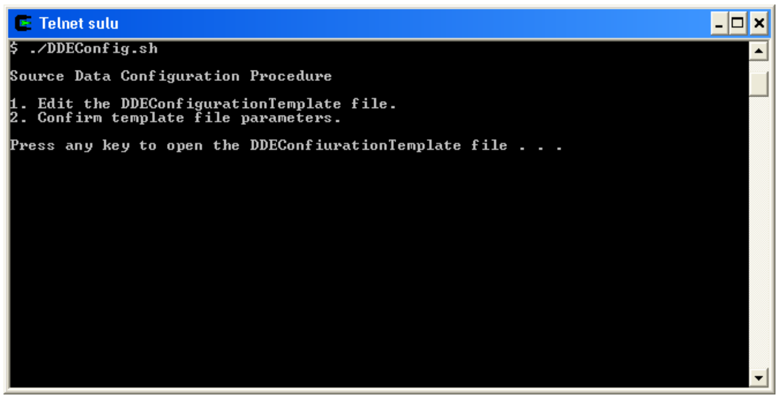
Once installed, Spatial Assistant can be run on the source data to create the `config.csv` file, which can then be copied to the target UNIX system running DDE.

Alternatively, UNIX users can edit `config.csv` using any text editor.

2 **Execute DDEConfig.sh.**

`DDEConfig.sh` is located in `<DDEInstallDir>/config/sourceDefiner.`

Execute the `DDEConfig.sh` script using the command `./DDEConfig.sh` and the initial startup screen will be displayed, as shown below.

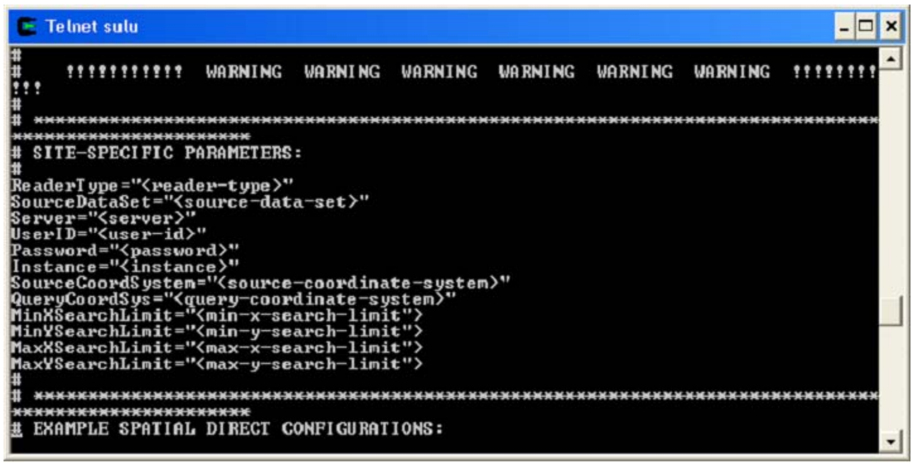


```
$ ./DDEConfig.sh

Source Data Configuration Procedure

1. Edit the DDEConfigurationTemplate file.
2. Confirm template file parameters.

Press any key to open the DDEConfiurationTemplate file . . .
```

3 Edit the DDEConfigurationTemplate file.

Press **Enter** to open the `DDEConfigurationTemplate` file with the vi editor (or the environment-specified editor).

Scroll down to the SITE-SPECIFIC PARAMETERS section to define your source data.



```
#
#    !!!!!!!!!!!  WARNING  WARNING  WARNING  WARNING  WARNING  WARNING  !!!!!!!
!!!
#
# ************************************************************************************
***********************
# SITE-SPECIFIC PARAMETERS:
#
ReaderType="<reader-type>"
SourceDataSet="<source-data-set>"
Server="<server>"
UserID="<user-id>"
Password="<password>"
Instance="<instance>"
SourceCoordSystem="<source-coordinate-system>"
QueryCoordSys="<query-coordinate-system>"
MinXSearchLimit="<min-x-search-limit">
MinYSearchLimit="<min-y-search-limit">
MaxXSearchLimit="<max-x-search-limit">
MaxYSearchLimit="<max-y-search-limit">
#
# ************************************************************************************
***********************
# EXAMPLE SPATIAL DIRECT CONFIGURATIONS:
```

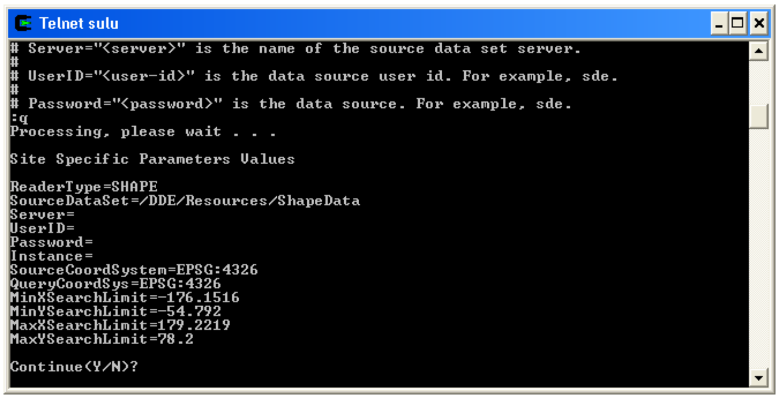
Parameter definitions are provided at the start of the `DDEConfigurationTemplate` file and example DDE Configurations are provided for both `SHAPE` and `SDE30 ReaderTypes` below the Parameters section.

4 Exit the editor.

Once you have defined the source data parameters, save the file and exit the editor to continue DDE configuration.

The `DDEConfig.sh` script processes the `DDEConfigurationTemplate` file and displays the result for confirmation.

- To re-edit the `DDEConfigurationTemplate` file, type **N** to quit and then re-run `DDEConfig.sh`.



```
# Server="<server>" is the name of the source data set server.
#
# UserID="<user-id>" is the data source user id. For example, sde.
#
# Password="<password>" is the data source. For example, sde.
:q
Processing, please wait . . .

Site Specific Parameters Values

ReaderType=SHAPE
SourceDataSet=/DDE/Resources/ShapeData
Server=
UserID=
Password=
Instance=
SourceCoordSystem=EPSG:4326
QueryCoordSys=EPSG:4326
MinXSearchLimit=-176.1516
MinYSearchLimit=-54.792
MaxXSearchLimit=179.2219
MaxYSearchLimit=78.2

Continue(Y/N)?
```

- To continue the script processing and complete the DDE source data configuration, type **Y**.

A notification message will be displayed once processing is complete.

DDE has now been configured to recognize your user source data but further configuration steps are required in order to integrate DDE with ArcIMS.

To complete the integration process, please continue to *Integrating DDE with ArcIMS* on page 81.

CHAPTER 6

Integrating DDE with ArcIMS

Pre-Integration Requirements

Before attempting to integrate DDE with ArcIMS, ensure that both ArcIMS and DDE have been installed and tested in their stand-alone configurations.

You should also be familiar with their use and configuration and you must have administrator access to ArcIMS and DDE, as well as to the web server and servlet engine applications that are installed on your system.

Warning

The following sections assume a basic knowledge of website configuration skills including familiarity with HTML and Javascript programming languages.

The sections also assume a standard ArcIMS HTML Viewer website configuration, as created using the ArcIMS administration tools. If your website differs significantly from this configuration, then it is advisable for the original website developer to perform the following steps of the integration.

Before You Begin

Before you begin the integration of the DDE with ArcIMS, the integration components must be located on the installation CD and extracted to a temporary location. The integration components are located at:

```
<CD>\DDE\DDEIntegration.zip
```

For the remainder of this document, your temporary extract location will be identified as *<tempDDE>*.

Web Server and Servlet Engine Configuration

This section describes the requirements for configuration of the web server and servlet engine but, due to the high number of possible web server and servlet engine combinations, does not give details of how this may be achieved. For specific configuration instructions for your particular web server and servlet engine please refer to the appropriate documentation supplied with the respective products.

Installing the DDE Components

- Copy the **directory** `<tempDDE>/ArcIMS/` to the `<DDEInstallDir>` directory.

Defining the Virtual Directory

Now define a virtual directory, `FME_ArcIMS`, in your web server which points to the directory which was created in the previous step.

`FME_ArcIMS`	This should point to your `<DDEInstallDir>/ArcIMS` directory.

Configuring the translationServlet

The default DDE installation makes use of Tomcat (Windows and Linux) or ServletRunner (Solaris, HP-UX, AIX) for running the `translationServlet`. If you would prefer to make use of your web server's native servlet engine or of an independent servlet engine (such as `ServletExec`), you will need to configure the servlet engine to recognize the `translationServlet` and to make use of the servlet properties described in *Servlet Properties* on page 202.

Once you have configured your servlet engine, there are two `translationServlet` properties that must be changed from their default values, which are `urlBase` and `htmlTemplatesDir`. Both the default and replacement values are defined in each of the servlet sections below.

Tomcat Servlet Engine: In the default Windows and Linux DDE configuration, the Tomcat properties file is located in:

```
<DDEInstallDir>/tomcat/webapps/servlet/WEB-INF/web.xml
```

Note: If Microsoft IIS is used as the web server, than your `cgi-bin` directory will likely be named "Scripts". If necessary, you may also need to modify the `servletURL` parameter to reflect this.

The default and replacement values for `urlBase` and `htmlTemplatesDir` are defined below.

Replace:

```
<init-param> <param-name>urlBase</param-name>  <param-value>http://
<localhost>:<port>/safeViewerHTML</param-value> </init-param>
<init-param> <param-name>htmlTemplatesDir</param-name>  <param-
value><web_server_document_root>/safeViewerHTML/
htmlTemplates</param-value> </init-param>
```

with:

```
<init-param> <param-name>urlBase</param-name>  <param-
value>http://<localhost>:<port>/FME_ArcIMS/safeViewerHTML</param-
value> </init-param>
<init-param> <param-name>htmlTemplatesDir</param-name>  <param-
value><DDEInstallDir>/ArcIMS/safeViewerHTML/
htmlTemplates</param-value> </init-param>
```

ServletRunner Servlet Engine: In the default Solaris, HP-UX and AIX DDE configuration, the `ServletRunner` properties file is located in:

```
<DDEInstallDir>/Jsdk/webpages/WEB-INF/servlets.properties
```

Replace this value:

```
urlBase=http://<localhost>:<port>/safeViewerHTML,\
htmlTemplatesDir=<web_server_document_root>/safeViewerHTML/
htmlTemplates,\
```

with this value:

```
urlBase=http://<localhost>:<port>/FME_ArcIMS/safeViewerHTML,\
htmlTemplatesDir=<DDEInstallDir>/ArcIMS/safeViewerHTML/
htmlTemplates,\
```

CHAPTER 7

Enabling an ArcIMS Map for Translations

Each ArcIMS map that you wish to enable for use with DDE must be configured individually. This is required because each map defined by ArcIMS resides in its own directory structure with its own map files. This chapter defines the customization process required for enabling an ArcIMS map for translations.

Wherever possible, an attempt has been made to describe a generic approach to this customization process; however, due to the highly flexible and configurable nature of defining ArcIMS web maps, the details outlined below may differ from your actual installation.

The steps that follow attempt to:

- add a new button to the ArcIMS map toolbar;
- configure the button so that, when clicked, it opens the `fmeDownload.htm` file in a new browser window.

If your ArcIMS website design differs significantly from the standard website that is created by the ArcIMS administration utilities, then it is suggested that the original web site developer perform the integration.

Conventions Used in this Chapter

`<tempDDE>`	should be interpreted as the directory in which the zip file was unzipped. e.g. `<tempDDE>\DDE\DDEIntegration.zip`

`<ArcIMSData>`	should be interpreted as a reference to your ArcIMS website data root directory, e.g.:
	`NT - C:\ArcIMS`
	`Solaris - /export/home/arcims/workdir`

`<ArcIMSMap>`	should be interpreted as a reference to your ArcIMS website data map directory. This will be different for each ArcIMS map but will have the form: `<ArcIMSData>/Website/map_name`
`<DDEInstallDir>`	should be interpreted as a reference to your DDE installation directory.

Modifying the Map Configuration Files

In order to enable the DDE functionality from your ArcIMS map, a new icon must be added to the map window toolbar and some of the map configuration files must be modified to enable that icon. The steps involved in this process are as follows:

1. Open the file `<ArcIMSMap>/ArcIMSparam.js` for editing. At the end of the variable declaration section containing similar `var use???=true;` statements, add the line:
   ```
   var useFMEExport=true;
   ```
2. Copy the `<tempDDE>/website/FME` directory to the `<ArcIMSMap>` directory.
3. Edit `<ArcIMSMap>/FME/fmeDownload.htm` as follows:
   - Replace all occurrences of `<localhost>` with your local host's name.
   - Replace `<cgiScriptRootDir>` with the name of the directory used by your web server for storing/running cgi scripts (usually `cgi-bin`).
   - Replace `<sdScriptDir>` with the name of the DDE scripts directory defined during installation (`DDE` by default).
   - If necessary, edit the ArcIMS Version Number section at the start of the file and update the `aims_version` variable to reflect the version of ArcIMS that you are using. **Note:** Currently only version 3.0 is processed differently – all other version numbers entered will be treated as equal.
   - Edit the LAYER TO THEME MAPPING DEFINITION section near the start of the file and define the `layerThemes` array. See step 4 below for details.
   - Edit the **Download Format Selection Box** section near the end of the file to remove any formats that you do not wish to make available to your end users for translations. The syntax for specifying an available format is:
     ```
     <OPTION VALUE="mapping_file_name"> format_description
     </OPTION>
     ```
     For the purpose of testing, we have included the GIF Image output option and made this the default selection. It is most likely that you will

want to remove this format from the list and to make an alternative format the default selection.

- To define the default selection simply use the `SELECTED` keyword after the option value. For example:

```
<OPTION VALUE="2e00.fme" SELECTED>ESRI Arc/Info Export
(E00)</OPTION>
```

- Edit the **Coordinate System Selection Box** section to define the available coordinate systems. Add or remove OPTION tags as necessary for the output coordinate systems which you wish to make available.
- The syntax for specifying an available coordinate system is:

```
<OPTION VALUE=”coord_sys_name”> description </OPTION>
```

- A list of all supported coordinate systems is available in `<DDEInstallDir>/fme/coordsys.db.`
- Edit the “Layer Selection Box” section to define the number of layers to be displayed in the layer select box on the download page.

  Warning: Internet Explorer versions 5.0 and earlier have a bug that causes the application to crash if the select box size is not large enough to contain all of the layer names. Make sure that your select box size is large enough to accommodate the complete list of layers available in your map.

4 In the `fmeDownload.htm` file, you must now define the ArcIMS layer to DDE theme name mappings which describe to DDE how to relate the layer, as displayed by an ArcIMS map, to the theme name that DDE has knowledge of.

- To identify your ArcIMS layer names, open the ArcIMS AXL file for the map you are configuring, and search for layer definition lines. For example, in the following layer definition line, the layer name is `"Art Galleries"`:

```
<LAYER type="featureclass" name="Art Galleries"
visible="true" id="artgalleries" maxscale="1:80000">
```

- To identify your DDE theme names open the `<DDEInstallDir>/translationControl/system/config.csv` file with any standard text editor. The `config.csv` file consists of multiple "columns" of data, delimited by the "|" separator character. The first column contains the theme name. For example, in the following partial `config.csv` entry, the theme name is `artgalleries^SHAPE_DATA`:

```
artgalleries^SHAPE_DATA|artgalleries|UNKNOWN|0|1|yes|
NAME char(38) FAX char(12) AV_SCORE decimal(6,0)...
```

Note The config.csv file is created and maintained under normal operation by the DDE SpatialAssistant application but for our purposes here, it is sufficient to view it using a standard text editor. For more information on using SpatialAssistant please refer to *Creation and Modification of config.csv*, on page 108.

- For each layer in your ArcIMS map for which DDE download is to be enabled, a layer-to-theme name mapping must be defined. To do this, add a line to the Layer to Theme Mapping Definition array at the start of the `fmeDownload.htm` file. The layer to theme mapping line has the format:

  ```
  layerThemes["My ArcIMS Layer Name"] =
  "DDEThemeName^DDE_DATA_SOURCE";
  ```

  For example, using the art galleries examples described above, the corresponding layer to theme mapping line would be:

  ```
  layerThemes["Art Galleries"] =
  "artgalleries^SHAPE_DATA";
  ```

Note The ArcIMS layer names and the DDE theme names supplied in the layer to theme mapping lines must match EXACTLY the values defined in the AXL and config.csv files, respectively. This includes all capitalization, spaces and other punctuation characters.

Once all layer-to-theme definitions have been completed, save your changes to the `fmeDownload.htm` file, and exit the file.

5 Copy the file `<tempDDE>/website/javascript/fme_export.js` to the `<ArcIMSMap>/javascript` directory.

6 Edit `<ArcIMSMap>/javascript/fme_export.js` as follows:

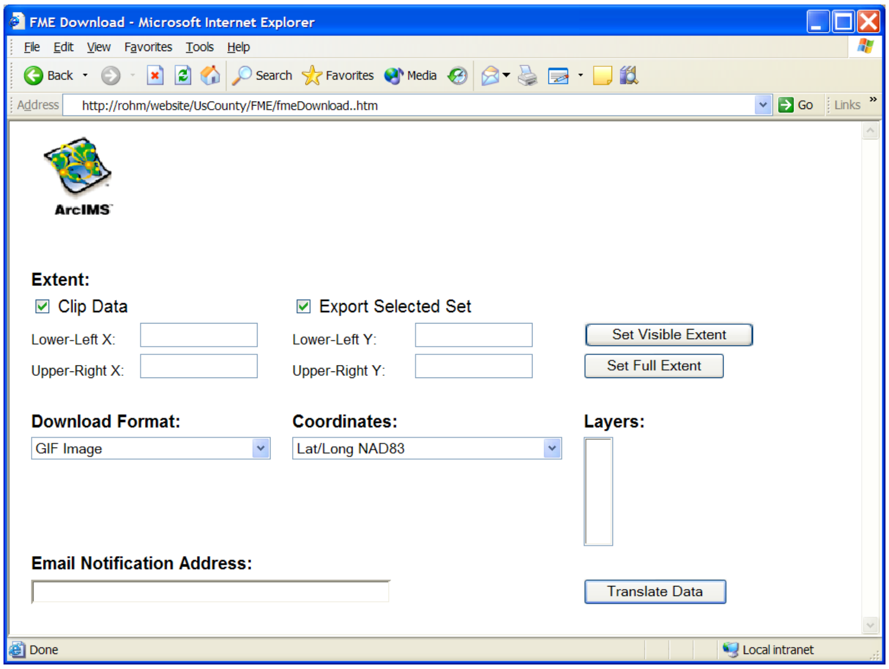
- Replace the string `<localhost>` with the host name of your website computer.
- Replace the string `<website>` with the directory name for this website. (Note that it is only the directory **name** that is required, not the full directory path.) For example: If your computer hostname is “rohm” and your ArcIMS website is named `UsCounty`, the line:

  ```
  var fmeWin =
  window.open("http://<localhost>/website/<website>/FME/fmeDown-
  load.htm","FMEDownload","width=750,height=500,toolbar=yes,sta-
  tus=yes,scrollbars=yes,resizable=yes");
  ```

  would become:

  ```
  var fmeWin =
  window.open("http://rohm/website/UsCounty/FME/fmeDownload.htm","FME-
  Download","width=750,height=500,toolbar=yes,status=yes,scroll-
  bars=yes,resizable=yes ");
  ```

- Verify that the URL you have just defined in the `window.open()` statement points to a valid location for your system.
- Open a web browser and, in the address field, enter the URL defined for the `window.open()` statement above. For example, attempt to open the page `http://rohm/website/UsCounty/FME/fmeDownload.htm.` The DDE order form page should be displayed (but the Extent coordinate fields and the Layers select box will be empty).


FME Download - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
ArcIMS
Done
Local intranet

http://rohm/website/UsCounty/FME/fmeDownload..htm

Extent:

☑ Clip Data ☑ Export Selected Set

Lower-Left X: Lower-Left Y: Set Visible Extent

Upper-Right X: Upper-Right Y: Set Full Extent

Download Format: GIF Image

Coordinates: Lat/Long NAD83

Layers:

Email Notification Address:

Translate Data

- If the above order form page is not displayed, check that the URL you have entered is valid for your web server and that the file exists.
- Save and exit `<ArcIMSMap>/javascript/fme_export.js`.

7 Open the following two files for editing:
`<tempDDE>/website/fme_toolbar.htm`
`<ArcIMSMap>/toolbar.htm`

8 Search for the section that defines multiple blocks of statements defining the toolbar buttons, such as:

```
.
.
.
if (parent.MapFrame.useZoomIn) {
  // Zoom In . . . requires aimsNavigation.js
  document.write('<td align="center" valign="middle">');
  document.write('<img src="images/zoomin_1.gif" width=16 height=16
hspace=1 vspace=0 border=0 alt=" ' + t.buttonList[16] + '" name="zoomin"
onmousedown="parent.MapFrame.clickFunction(\'zoomin\'); setToolPic(\
'Zoom In\');"
onmouseover="window.status=\' ' + t.buttonList[16] + '\'">');
  isSecond = !isSecond;
  document.writeln('</td>');
  if (isSecond) document.write('</tr><tr>');
}
.
.
.
```

9 Insert the text from the `<tempDDE>/ArcIMS/website/fme_toolbar.htm` file as a final block in this set of "if" statements so that the `<ArcIMSMap>/toolbar.htm` file now looks like:

```
.
.
.
if (parent.MapFrame.useFMEExport) {
document.write('<td align="center" valign="middle">');
document.write('<img src="/FME_ArcIMS/safeViewerHTML/images/
SpatialDirectButton.gif" width=16 height=16 hspace=1 vspace=1
border=0
alt="Export Map Using DDE" onmousedown="parent.MapFrame.
fmeExport();" onmouseover="window.status=\'Translate and Download
Map Using
DDE\'">');
isSecond = !isSecond;
document.writeln('</td>');
if (isSecond) document.write('</tr><tr>');
}
document.writeln('</tr>');
document.writeln('</TABLE>');
</script>
.
.
.
```

- Save and exit `<ArcIMSMap>/toolbar.htm`.

10 Edit `<ArcIMSMap>/MapFrame.htm`. At the end of the block of similar Javascript source definition statements, add this line:

```
<SCRIPT TYPE="text/javascript" LANGUAGE="JavaScript"
SRC="javascript/fme_export.js"></SCRIPT>
```

- Save and exit `<ArcIMSMap>/MapFrame.htm`.

At this point the DDE integration with ArcIMS should be complete. To test the integration, please proceed to *Using DDE for ArcIMS*, on page 91.

CHAPTER 8

Using DDE for ArcIMS

Running DDE

Before you can use DDE for ArcIMS you must ensure that DDE is running and accepting translation requests.

Windows

If you have configured DDE to run as a service, check that it is running through the Services icon in the Control Panel. If you have not configured DDE as a service, then you can start it by clicking **Start | Programs | DDE | Start DDE** from the Windows task bar.

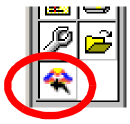
UNIX

To start DDE, run the `startSD` shell script located in the `<DDEInstallDir>`.

Performing a Translation

1 Start your web browser and enter the URL of your map website to bring up the initial page.

2 Zoom in to your desired area and, optionally, select a feature set for export. Once your desired view has been obtained, click the **Export Map Using** DDE button at the bottom of the toolbar to the left of the screen.



3 The DDE order form will appear and will reflect the contents of your ArcIMS session at the time the DDE button was pressed. That is, the current map extent will be populated in the extent boxes, the currently visible layers

will be highlighted in the Layers box or a selected set will have been identified as the layer available for export.

Note: Selected set processing is not available for Shapefile source datasets.

4 Select your download format and coordinate system, then press the **Translate Data** button to start the translation and wait for the translation results window to appear.

Your translation is now complete. Click the links to download the data for use as required.

CHAPTER 9

Customizing the FMEExport Function

The functionality of the **DDE** button can be fully customized to suit the requirements of your organization. You can do anything from changing the look and feel of the Order Form, to bypassing this form entirely and allowing a “one click” export from the ArcIMS window.

Customization is left to the client’s discretion and the functionality currently provided should serve only as a guideline for further development. A detailed explanation of this customization is beyond the scope of this document and the client is referred to the appropriate literature concerning HTML and JavaScript programming.

It should be noted that, for the majority of cases, the only file that would need to be modified for customization is

```
<ArcIMSMap>/Fme/fmeDownload.htm
```

Part 3: Customizing DDE

CHAPTER

10

Source Data Theme Definition

In standard configurations, DDE is made aware of its source data by defining each source theme[1] as an entry in a file called `<DDEInstallDir>\translationControl\system\config.csv`. This file is the primary configuration mechanism for defining source data to DDE.

The `config.csv` file is an ASCII text file whose rows are delimited into fields using the pipe character (|). Each row represents a source data theme and each field represents a parameter affecting some aspect of how the FME Server or the Translation Servlet treats the presentation of the theme. This chapter describes each of the fields contained in `config.csv`.

The FME Server and the Translation Servlet components both use the information in `config.csv`. The descriptions below indicate which component uses which field.

In general, value modifications of fields used by the Translation Servlet require restarting DDE for the new values to take effect, whereas modifying values of fields used by the FME Server do not require a DDE restart.

Fields in *config.csv*

The following fields are defined for each theme in `config.csv`.

Theme Name

The `Theme Name` is the name of the source data theme and acts as the main lookup key into the file. A theme will usually consist of a collection of data with similar geometry and attributes and is largely dependent on the source dataset. For example, with Shape, each theme is equivalent to a layer, while with SDE,

1. In DDE, *theme* is used as a generic term which can be taken to mean *layer*, *table* or *level*, depending on the source data format in use.

each theme corresponds to a spatial table. ***Used by FME Server and Translation Servlet.***

Theme Groups

Optionally, two or more theme names can appear in a theme name field to form a *theme name group*. Each component theme in the group must already be fully defined with a separate entry in the same `config.csv` file.

Each theme name in the group must be separated from adjacent names within the field by a space. Theme definitions whose first field is a theme name group require only the `Web Name` and `Web Category` fields (second and third fields respectively described below) to be specified. None of the remaining fields in the `config.csv` file are relevant for theme name group entries and should not be specified.

Theme definitions of this type are used only by the Translation Servlet when it generates the theme name pull-down lists for the default order form. Theme entries defined as groups are ignored by the FME Server.

When the Servlet encounters a theme name group entry, it generates a single item for that entry in the theme pull-down list, and associates the theme names in the group with that single item. When the item is selected, it has the effect of selecting all the individual theme names in the associated theme group. This acts as a convenient way of selecting multiple themes with a single choice.

A theme name can be present in more than one theme name group. If selection of multiple groups causes the same theme name to be selected more than once, the Servlet automatically removes the duplicate selections.

Because each theme whose name appears in a group must already be fully defined with a separate entry, by default it will also appear on its own as a separate item in the theme pull-down list. This will be in addition to its implicit presence in the list as part of a selectable theme group.

To prevent a theme name that is part of a group from appearing on its own as a separate item, locate the theme's separate definition entry and set the value in its `Web Name` and/or its `Web Category` field to be either a "-" hyphen character or the word `"delete"`. Doing this does not alter the theme's definition in any way – it merely prevents the theme's name from appearing as a separately selectable item in the pull-down list.

Theme Name Groups and Web Categories: A theme name group is different from a web category (described below). A web category is simply a collection of individually selectable theme items displayed within a single pull-down list in the default order form. Each item within the list can represent either a single theme or a theme name group. Categories are used to gather and display theme names into functionally related collections such as *Hydrography*, *Transportation* or *Vegetation*. Theme name groups are used as a convenience mechanism to select multiple themes via a single item selection within the category list.

Example: Here is an example of three theme definition entries that use theme name groups as they would appear in the `config.csv` file. Note that only the first three fields are specified:

```
airports helipads runways|Air|Transportation
roads railways trails|Ground|Transportation
harbors ports canals locks|Water|Transportation
```

These entries would cause a category pull-down list called `Transportation` to be displayed in the default order form with `Air`, `Ground` and `Water` items appearing in the list. Selecting one of these items would result in the actual selection of the themes defined in the first field associated with that item selection. For example, selecting the `Ground` item in the list would cause the `roads, railways` and `trails` themes to become selected for translation.

Web Name

`Web Name` is a user-visible descriptive name for the `THEME NAME` and appears in the theme selection lists in the default DDE order form web page. If the actual theme name is terse, such as `FORPOLY`, then the web name for the theme can be set to “Forest Cover Polygon”. A user interacting with the browser would see this more descriptive name. The Translation Servlet maps this name to the associated `Theme Name`. If this field’s value is either a “-” hyphen character or the word `"delete"`, the theme’s name will not appear in the theme pull-down list in the default order form. This field is only relevant when the DDE HTML front-end application is being used. ***Used by Translation Servlet.***

Web Category

`Web Category` determines which browser pull-down selection list displays the theme’s web name. In the default DDE order form web page, there are a number of pull-down lists. The Servlet automatically creates a pull-down list for each different web category name it encounters and displays each theme’s `WEB NAME` in the appropriate list. The list itself is labelled using the category name. Spatial Assistant’s default action is to place all themes in a single category pull-down list called “unknown”. This is a convenient method for creating a collection of themes for presentation to a browser user. If this field’s value is either a “-” hyphen character or the word “`delete`”, the theme’s name will not appear in the theme pull-down list in the default order form. This field is only relevant if the DDE HTML front-end application is being used. ***Used by Translation Servlet.***

Min Visibility Area/Max Visibility Area

The minimum and maximum visibility fields are relevant if the DDE HTML front-end application is being used. These fields support simple scale-based theme selection. Note that these two fields have an effect only when the `enableAutoThemeSelection` property used by the Translation Servlet is set to `true`. (This property is discussed *Property Descriptions* on page 205.)

The two visibility area factor fields represent the min and max areas within which the theme will be automatically selected and therefore present in the translated output (assuming auto theme selection is enabled and features are actually present in the theme for the area in question).

The field values aren't actual ground-unit areas but rather multiplier factors between 0.0 and 1.0 inclusive. The Servlet computes the maximum source data area (obtained from the `minXSearchLimit, maxXSearchLimit, minYSearchLimit and maxSearchLimit` Servlet properties) and then multiplies this by the min and max visibility area factors for each theme. The result is a min and a max area limit for each theme. During a translation, the current extent area is compared to each theme's area limits. If the current viewing area falls between the theme's area limits then that theme is automatically selected for output. If a theme's area limits do not bracket the current viewing area then the theme is not selected.

Appropriate adjustment of each theme's visibility area factors can create the effect of themes dropping in and out depending on the viewing area in effect at the time of each translation. This effect is most visually noticeable when zooming on a displayable output such as GIF.

For example, major highways could be set with a minimum and maximum visibility of 0.0 and 1.0 respectively. As a result they would always be displayed or included in a translation. Minor highways could be set to 0.0 to 0.5 and all urban roads set to 0.0 to 0.2. In this scenario when a user is viewing a large scale area, they would see all road and highway types, and as they zoomed out the urban roads, and finally the minor roads would no longer be displayed. ***Used by Translation Servlet.***

Initial Display

This field represents a flag indicating whether or not the theme should appear by default and is relevant if the DDE HTML front-end application is being used. This flag is effective only when the `enableAutoThemeSelection` Servlet property is set to `false`.

The `Initial Display` value should appear as the string `yes` or `no`. A string of `yes` causes the theme to be displayed by default unless it has been explicitly deselected; any other value prevents the theme from appearing by default *unless* auto-theme selection is in effect. In this case if the current viewing area is within the visibility limits of a given theme, then the theme is displayed. ***Used by Translation Servlet.***

Attributes

This field contains the name and data type of each attribute for the theme. The attribute definition information in this field tells the FME Server which attributes it should include for the theme, enabling it to output them to formats that support attributes. If attributes are available and desired, they should be specified this field. Only those attributes desired for output need be present. All, some, or none can be included.

Attribute definitions have the following syntax:

```
    <name> <type> [<name> <type> ...]
```

where `<name>` is the name of the attribute and `<type>` specifies the data type of the attribute. When more than one attribute is defined, the `<name> <type>` pairs of each are entered one after the other in the field, separated by spaces.

The following attribute data type values can be used for `<type>`:

Data type	Description
`char(<width>)`	character string whose maximum length is defined by `<width>`.
`decimal(<width>, <decimal>)`	floating point value that is `<width>` places wide, including the decimal point) with `<decimal>` places after the decimal point.
`real32`	32-bit real number
`real64`	64-bit real number
`int16`	16-bit integer value.
`int32`	32-bit integer
`Boolean`	logical value. (true, false, yes, no)
`date`	date field of the form YYYYMMDD
`blob`	binary attribute. Attribute can be used to store anything from files to sounds to video.

Here is an example of an `Attributes` field entry defining several attributes:

```
RoadID char(5) NumLanes int16 PavementClass char(10)
RoadWidth decimal(10,2) Closed Boolean DateBuilt date
```

Note DDE automatically modifies output attribute names as required to conform to any syntax limitations placed on them by the output format, while attempting to keep them unique. In some cases, the latter is accomplished by appending an integer suffix to a truncated version of the name.

For example, in a Shape format file, attribute name lengths are constrained to be less than or equal to 10 characters, all in uppercase. If the source data had an attribute called TaxLotRegionValue then it would be changed to TAXLOTR<n> in the Shape file. See the *FME Readers and Writers* manual for details on particular formats.

The Spatial Assistant utility determines all attributes for each theme in the source data and creates the corresponding `<name> <type>` attribute definitions in the `ATTRIBUTES` field. ***Used by FME Server.***

Legend Entry

This field contains a string indicating the name of the theme as displayed in the legend present in the default DDE GIF output web page. As such, it is relevant only for the DDE HTML front-end application. ***Used by FME Server.***

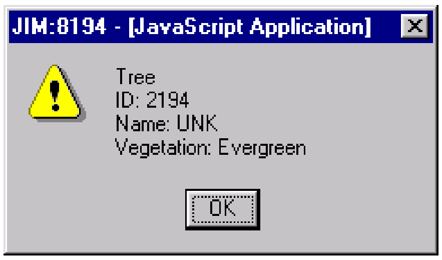
GIF Attrs

This attribute is relevant for the DDE HTML front-end application. In the default DDE GIF output web page, the generated GIF (also known as an intelligent GIF) allows a user to click on a displayed feature and have the attributes of that feature be displayed in a pop-up panel.

The content of this panel is determined by the string value of the `GIF Attrs` field. The string can be any text, but will usually include attribute names and references to their values using the “&” value-of operator. This will cause the panel to display the feature’s attributes and its associated values. Any of the attributes contained in the `ATTRIBUTES` field can be included in this string.

An example `GIF Attrs` string and the pop-up panel contents it generates are shown below:

```
Tree\nID: &ID \nName: &NAM \nVegetation: &VEGdesc
```


JIM:8194 - [JavaScript Application]

Tree
ID: 2194
Name: UNK
Vegetation: Evergreen

Note the use of the `\n` to force a new line. Care should be taken to always include at least one space after each `&` reference to an attribute value. If this is not done, the system will not recognize the name as referring to an attribute. For example, `ID: &ID \nName:` will display correctly but `ID: &ID\nName:` will display incorrectly.

The Spatial Assistant utility automatically sets this field with the appropriate string value that will display all attributes found in the source data. ***Used by FME Server.***

GIF Color

This field contains the descriptive name of the color to use in GIF display of data. GIF color names are defined in the file `<DDEInstallDir>\translationControl\fmi\gif_def.fmi`. ***Used by FME Server.***

GIF Brush Width

This field contains the GIF line width integer to use when drawing a line feature. ***Used by FME Server.***

Allow Poly Fill

This field turns GIF polygon fill on or off. The value must be `yes` or `no`. ***Used by FME Server.***

GIF Precedence

This field is an integer which defines the rendering order of themes in a GIF. The GIF is rendered using a standard painters or Z-buffer algorithm. The theme with the lowest `GIF Precedence` value is rendered first, the next lowest follows, until all themes have been rendered. ***Used by FME Server.***

GIF Alt Label

This field is relevant for the DDE HTML front-end application. The `GIF Alt Label` has two functions.

First, it represents a string which is displayed when the mouse hovers over the feature. For example if the value for the `GIF Alt Label` for a given theme is set to `Forest`, then this will appear when the mouse pauses over a region of a GIF where this feature is the top feature of the display. Not all browsers support this alternate label feature. Browsers that *do* support it include Netscape Navigator and MS Internet Explorer.

The second function of `GIF Alt Label` is to enable clickable image mapping for the theme. If the value of `GIF Alt Label` is "-" (a hyphen) then no image map is created for the theme. If the value is anything else, then an image map is created for the theme. See the description of `GIF Attrs` and *GIF Image Display Characteristics* on page 264 for details concerning clickable GIF image mapping. ***Used by FME Server.***

GIF Font

This field represents the name of the font to use in GIF display images of text. There are five built-in fonts that may be used:

- `gif_font_tiny`
- `gif_font_small`
- `gif_font_medium`
- `gif_font_large`
- `gif_font_giant`

In addition to the built-in fonts, there are several Hershey bitmap fonts distributed with the FME. Some of these fonts are not suited for label text. The currently available Hershey fonts are:

- `Gothice`
- `Gothicg`
- `Gothici`
- `Italicc`
- `Italict`
- `Romanc`
- `Romand`
- `Romanp`*
- `Romans`*

- `Romant`
- `Scriptc`
- `Scripts`

* best choices for label text

Used by FME Server.

GIF Symbol

This field represents the name of a GIF file whose image will be displayed in GIF output for *point* features.

If this column has no value (just two pipe delimiter characters with nothing in between) then no GIF image symbol is defined for point features for the theme on the row. However, if there is a name in this column referring to a GIF file, then that image will be used for the theme's point features. The name in this field must have the following form:

```
<filename>.gif
```

For example: `blueDot.gif`

All point feature GIF image symbol files must be located in the directory:

```
<DDEInstallDir>\graphics
```

Used by FME Server.

File Name

This field represents the base name to use for the theme when outputting it to files in formats such as ESRI Shape. This name is a character string and can be composed of alphanumeric characters and the underscore character (_). ***Used by FME Server.***

AutoCAD Layer Name

This field represents the name to give the theme as an AutoCAD layer when translating to AutoCAD format. ***Used by FME Server.***

IGDS Type

This field represents the level number on which the features from the theme will be placed when translating to IGDS Microstation design files. It is an integer in the range of 1 to 63. ***Used by FME Server.***

IGDS Color

This field represents the color to be assigned to all of the features of the given theme when translating to IGDS MicroStation design files. It is an integer in the range of 0 to 255. The MicroStation local color table will dictate the actual colors that the IGDS color index will map to. ***Used by FME Server.***

IGDS Style

This field represents the line style to be assigned to all features in the given theme when translating to IGDS MicroStation Design files. It is an integer in the range of 0 to 7. ***Used by FME Server.***

IGDS Weight

This field represents the line weight to be assigned to all features in the given theme when translating to IGDS Microstation design files. It is an integer in the range of 0 to 31. ***Used by FME Server.***

Mapinfo Colour

This field represents the colour name to be assigned to all features in the given theme when translating to a MapInfo file. The MapInfo colour names are defined in file `<DDEInstallDir>\translationControl\fmi\mapinfostyles.fmi`. ***Used by FME Server.***

Mapinfo Pen Pattern

This field represents the pen pattern name that is assigned to all features in a given theme when translating to a MapInfo file. The MapInfo pen pattern names are defined in the file `<DDEInstallDir>\translationControl\fmi\mapinfostyles.fmi`. ***Used by FME Server.***

Mapinfo Brush Pattern

This field represents the brush pattern name that is assigned to all linear features in the given theme when translating to a MapInfo file. The MapInfo brush pattern names are defined in file `<DDEInstallDir>\translationControl\fmi\mapinfostyles.fmi`. ***Used by FME Server.***

CDF Major Code

This field represents the CDF format attribute major code integer. ***Used by FME Server.***

CDF Minor Code

This field represents the CDF format attribute minor code integer. ***Used by FME Server.***

Zycor Culture Code

This field represents the Zycor ZMAP format culture code integer. ***Used by FME Server.***

Seisworks Short Name

This field represents the Seisworks format file base name for multi-file export. ***Used by FME Server.***

Seisworks Color

This field represents the Seisworks format color name. ***Used by FME Server.***

Text Attribute

This field represents the name of a feature attribute whose value will be used to create a new text feature. For example, if the Text Attribute column contains the value `name` for the Road theme and an input Road feature has an attribute called `name` which contains the value `Stephens Rd`, then, on translation, a new text feature will be created to label the original line feature with the text `Stephens Rd`. ***Used by FME Server.***

Text Rotation

This field represents the degrees of rotation for output text features, measured in degrees counter-clockwise from the horizontal. ***Used by FME Server.***

Text Height

This field represents the height of output text features, measured in ground units of the output coordinate system. For example, when outputting to an Albers projection in meters, a text height value of around 10,000 could be tried. Some experimentation is usually needed with this value to get the text size right. If the value is too small no text will be seen. This field is applicable only for Hershey fonts. If built-in GIF fonts are used, this value is ignored. ***Used by FME Server.***

Theme Weight

This field represents the numeric weight assigned to the theme. Theme weight is a relative measure of a theme’s feature complexity and is used in the computation of overall request complexity by the Translation Servlet.

Higher theme weight values indicate greater theme complexity. Theme weight values are arbitrary floating-point numbers with no restriction on range.

A theme’s weight in effect represents the cost of translating that theme relative to the translation cost of the other themes in the same source dataset. For example, a contour line theme having many features and/or vertices would usually have a higher weight value than a political boundary theme containing fewer features and/or vertices. ***Used by Translation Servlet.***

Creation and Modification of *config.csv*

In most cases, `config.csv` need only be created once in order to define the source data to DDE. Subsequent modifications to the file are necessary only if source data themes are added or deleted or if DDE’s presentation of them needs to be changed.

If any fields used by the Translation Servlet are modified, DDE must be restarted for the changes to take effect. Changes to fields used only by the FME Server do not require restarting of DDE – they will take effect immediately upon the next translation.

Although `config.csv` can be created and maintained manually with a text editor, these actions are more easily performed using a supplied utility called Spatial Assistant (currently available for Windows systems only).

UNIX-based DDE users can create `config.csv` either manually using a text editor, or, if a Windows system with DDE installed on it is available and has read access to the UNIX source dataset, by running Spatial Assistant on the Windows system and copying `config.csv` to the UNIX system.

Creating *config.csv* with Spatial Assistant

1 Rename the existing `config.csv` file before creating a new one so that current settings are retained in a backup file.

2 Start Spatial Assistant by double-clicking on the `spatialassistant.exe` file located in `<DDEInstallDir>/spatialAssistant`.

3 Select **File | New** to display the Specify Input Data Source dialog box.

4 Specify the source format and dataset.

5 If applicable, click on the Settings button to enter appropriate values for the source data, then click OK in the Settings panel. (Press the F1 key for detailed help on settings boxes.)

6 Click OK in the Specify Input Data Source dialog box.

Spatial Assistant then proceeds to read the specified source dataset, determines the themes available and their attributes, and displays the information in table form. This information will be saved in the `config.csv` file being created.

Spatial Assistant assigns a default value to some of the fields. Field values can be changed at this point if desired by double-clicking on a field's cell and modifying its contents. If the field's value is enumerable, a list of possible values is displayed.

7 Save the newly created `config.csv` file by selecting File | Save As and save the file as `<DDEInstallDir>\translationControl\system\config.csv`.

The `config.csv` file can also be saved using a different file name if there is a need to support multiple configurations. However, DDE will always look for and use the file named `config.csv` in the default directory.

8 Select File | Exit to quit Spatial Assistant.

Modifying *config.csv* with Spatial Assistant

Once the `config.csv` file has been created, it can be subsequently modified with the Spatial Assistant utility, if a Windows system with FME installed is available and has access to the source data.

Existing theme entries in the file can be modified or deleted, and new ones can be added. These maintenance operations can be performed with Spatial Assistant using the following procedures:

1 Start Spatial Assistant by double-clicking on the `spatialassistant.exe` file located in

```
<DDEInstallDir>/spatialAssistant
```

2 Select File | Open and specify the `config.csv` file to modify.

3 To *change* a field value, double-click in its cell and enter the new value.

4 To *delete* a theme entry, select the theme to delete by clicking on its row, then select File | Remove Schema.

5 To *add* one or more theme entries:

- Select File | Add Schema and specify the source format and dataset in the resulting dialog box.

- If applicable, click on the Settings button to enter appropriate values for the source data, then click OK in the Settings panel
- Click OK on the Specify Input Data Source dialog box.
- Spatial Assistant will read the selected theme(s) from the source data and add an entry to `config.csv` for each theme, setting default field values as necessary.
- When all entries have appeared, the default values can be changed to more appropriate ones if desired.

6 To *save* the changes, select File | Save As and save the modified file as `config.csv`.

7 Quit Spatial Assistant by selecting File | Exit.

Location of *config.csv*

The `config.csv` file supplies information to both the FME Server and to the Translation Servlet. It is read primarily by the FME Server and as such is located in:

```
<DDEInstallDir>/translationControl/system/config.csv
```

on the system running the FME Server.

Because the Translation Servlet also needs to access the information in `config.csv`, the Servlet must be told where to find this file. This is done by setting the value of a servlet property called `layerListConfURL` in the Servlet's properties file.

The value of this property is a URL pointing to `config.csv`. During startup, the Servlet uses this URL to access the file, download its contents and create a local copy for itself to use during subsequent operations.

Translation Servlet and FME Server on the Same System

If the Translation Servlet and the FME Server are installed on the same system, the installation process automatically sets the URL to:

```
file://<DDEInstallDir>/translationControl/system/config.csv
```

This URL is, in effect, a pathname to the locally-resident `config.csv` file used by the FME Server installed on the same system. In this case, the Servlet doesn't create a copy for itself – it simply uses the same file that the FME Server uses.

Translation Servlet and FME Server on Different Systems

If the Translation Servlet and the FME Server are installed on different systems, the installation proceeds under a different assumption.

The assumption is that an intermediate copy of `config.csv` will be placed in a location that is accessible via a web or FTP server and that the URL used by the Servlet will reference that location to accomplish the copying.

It is further assumed that the location will be within the server's publicly accessible root directory. *This has security risk implications which are discussed below.*

Under this assumption, the FME Server support installation creates an empty directory called `spatialDirectConfigFileCopy` within the indicated web/FTP server's public root directory. The person responsible for installing DDE must then put an intermediate copy of the `config.csv` file into this directory to make the file accessible to the Servlet.

On the other side, the Translation Servlet installation will set the URL property value to:

```
<protocol>://<FMEServerHost>/spatialDirectConfigFileCopy/
config.csv
```

Upon startup, the Servlet accesses the file through the URL, downloads its content and makes a copy for itself. The Servlet creates this local copy in:

```
<DDEInstallDir>/safeViewerHTML/supportData/localConfig.csv
```

The directory path used by the Servlet for this copy is set by the servlet property `supportDataDir`.

Security Risk Implications

Warning

If a user puts an intermediate copy of `config.csv` into the web/FTP server's `spatialDirectConfigFileCopy` subdirectory on the FME Server system, it puts the file into a location from which it could possibly be viewed by outside users with browsers, thereby creating a potential security risk.

There is a better, more secure procedure that keeps `config.csv` out of public server directories while still allowing the Servlet access to it.

Instead of copying `config.csv` to the web/FTP `spatialDirectConfigFileCopy` subdirectory, it should instead be copied directly to:

```
<DDEInstallDir>/safeViewerHTML/supportData/localConfig.csv
```

on the Servlet system. The Servlet's properties file should then be edited to set the value of the `layerListConfURL` property to point to the local copy. The `layerListConfURL` property is changed to refer to the local copy by setting its value to:

```
file://<DDEInstallDir>/safeViewerHTML/supportData/
localConfig.csv
```

Upon start-up, the Servlet will now access the local copy of `config.csv` located in the directory specified above, rather than in the publicly accessible web/FTP server directory.

CHAPTER 11

Configuring Source Data Manually

Manual configuration is required to configure DDE to read from more than a single source dataset. This chapter describes how to manually configure DDE.

Configuration Steps

The main steps involved in configuration are:

1. Prepare for configuration.
2. Define the source data themes.
3. Specify the query coordinate system.
4. Specify the source data spatial extent.
5. Specify the output formats to be made available.
6. Specify the output coordinate systems to be made available.
7. Specify source data format.
8. Restart DDE.

The following sections describe each of these steps.

Prepare for Configuration

As distributed, DDE is initially configured to access a supplied sample source dataset. Before configuring DDE to read a different source dataset, perform the following steps:

- Ensure that DDE has been successfully installed and run with the sample source dataset provided.
- Shut down DDE before configuring for a new source dataset.
- Ensure that the source dataset is correctly set up, and that its spatial extent and coordinate system are known.

Define Source Data Themes

Defining source data themes for DDE involves creating a primary configuration file, defining data source keywords and setting theme weights. These procedures are described in the following sections.

Create the config.csv File

In the default configuration, DDE is made aware of its source data by defining each source theme as an entry in a file called `<DDEInstallDir>\translationControl\system\config.csv`. This file is the primary configuration mechanism for defining source data to DDE. *Source Data Theme Definition* on page 97 gives a detailed description of `config.csv`.

The first step in configuring to read a source dataset is to create the `config.csv` file and populate it with entries defining the source data themes.

Using Spatial Assistant

Although `config.csv` can be created and maintained manually with a text editor, these actions are more easily performed using a supplied utility called Spatial Assistant.

Spatial Assistant is currently available for Windows systems only. UNIX-based DDE users can create `config.csv` either manually using a text editor, or, if a Windows system with FME installed on it is available and has read access to the UNIX source dataset, users can run Spatial Assistant, create `config.csv` and copy it to their UNIX system.

Creation and modification of `config.csv` with Spatial Assistant is discussed in *Creation and Modification of config.csv* on page 108.

Data Source Keywords

Once the `config.csv` file has been created, a data source keyword must be appended as a suffix to the end of the theme names in the first column.

Each appended suffix identifies the data source from which the theme originates. The suffixes also enable theme names to be distinguished from one another in cases where themes originating from multiple sources have the same root name.

Note: If themes originating from multiple sources have the same root name, the names in the "File Name" field in `config.csv` (17 th field in the file) should also be modified such that each name is unique as well. The keyword suffixes can be used for this purpose if desired, but any naming scheme can be used as long as it results in each theme entry having a unique "File Name" field value.

Note: All source data is defined in DDE using the multi-source mechanism. This is true even when only a single data source is being used. In this case it simply means that the source environment is defined as a multi-source configuration consisting of a single data source.

By default, each data source suffix consists of a "^" caret character, followed by the keyword of the data source.

Data source keywords are chosen by the user to reflect the characteristics of each data source. Each data source keyword must be unique within a DDE configuration and cannot contain any spaces.

The keywords used here must be the same ones used when defining data source mapping files, as described in *Specify Source Data Formats* on page 121.

Keywords can be names such as *municipal*, *state*, *federal*, *industrial*, *commercial* or any other term that uniquely identifies the data source. In general it is preferable to choose keywords that reflect the nature, origin or function of the source data rather than the format that the data is stored in.

Example

Suppose two data sources are being used. The first source is provided by a federal agency and contains two themes named `Trees` and `Grass`. The second source is provided by a commercial firm and contains two themes named `SDE.POPULATION` and `SDE.CONTOURS`.

The keyword of the first data source is chosen to be `federal` and the second to be `commercial`.

After theme name modification, each theme's name in the first column becomes:

```
Trees_federal
Grass_federal
```

```
SDE.POPULATION_commercial
SDE.CONTOURS_commercial
```

After applying the modification the following sample entries would exist in `config.csv` (the form shown is the raw ASCII text seen when the file is viewed with a text editor, not with Spatial Assistant):

```
Trees_federal|Trees|Vegetation|0|1|yes|NAM char(50) VEGdesc
char(50) ID char(10)|Trees|Trees\nNAM: &NAM \nVEGdesc:
&VEGdesc \nID: &ID \n|GrassGreen|1|yes|1|Trees|
gif_font_large||Trees|TREES|1|0|0|0|Black|wideDashed|
rightDiag|0|0|0|x|BLACK||0|10|1.0

Grass_federal|Grass|Vegetation|0|1|yes|F_CODEdesc char(50)
ID char(10)|Grass|Grass\nF_CODEdesc: &F_CODEdesc \nID: &ID \
n|BrightGreen|1|yes|2|Grass|gif_font_giant||Grass|GRASS|2|1|
1|1|CobaltBlue|wideDashed|horizontal|0|0|0|x|BLACK||1|10|1.0

SDE.POPULATION_commercial|Secondary Towns|Population|0|1|
yes|TXT char(50) ID char(10)|Secondary Towns|SDE.POPULATION\
nTXT: &TXT \nID: &ID \n|MediumCyan|1|yes|4|SDE.POPULATION|
Romand||SDE.POPULATION|SDE.POPULATION|1|0|0|0|BrightMagenta|
solidPen|rightDiag|0|0|0|x|BLACK||0|10|1.0

SDE.CONTOURS_commercial|Contours (sparse)|Boundaries|0|1|
yes|ZV2 char(10) ID char(10)|Contours (sparse)|SDE.CONTOURS\
nZV2: &ZV2 \nID: &ID \n|DarkGray|1|yes|0|SDE.CONTOURS|
Romand||SDE.CONTOURS|SDE.CONTOURS|1|0|0|0|Brown|dotted|
solid|0|0|0|x|BLACK||0|10|1.0
```

Changing the Default Suffix Delimiter Character

You can change the default suffix delimiter character. One reason for wanting to do this might be to avoid confusion when theme names contain an underscore (_) character as a valid part of the theme name itself. The example below changes the keyword suffix delimiter character from the default underscore (_) to the caret (^) character.

1 Edit the file `<DDEInstallDir>\translationControl\importers\multiSource.fmi` as follows:

- Modify the line:

  ```
  MACRO sourceKeywordDelimiter _
  ```

  by replacing the "_" with the new delimiter character. For example:

  ```
  MACRO sourceKeywordDelimiter ^
  ```

- Save the file.

2 Edit the file `<DDEInstallDir>\translationControl\system\config.csv` as follows:

- Every occurrence of each theme's suffixed name needs to be updated to use the new delimiter character. For example:

  ```
  COASTL_sample|Coastlines|Boundaries|0|1|yes|ID
  char(10) |Coastline|......
  ```

  would become:

  ```
  COASTL^sample|Coastlines|Boundaries|0|1|yes|ID
  char(10) |Coastline|......
  ```

Source Theme Weight

The Theme Weight field in the `config.csv` file allows a numeric weight value to be specified for each theme.

Theme weight is a relative measure of a theme's feature complexity and is used in the computation of overall request complexity by the Translation Servlet. This computation is only performed when translation request filtering is enabled (it is disabled by default). *Translation Request Filtering* on page 220 describes request filtering and complexity calculation.

Higher theme weight values indicate greater theme complexity. Theme weight values are arbitrary floating-point numbers with no restriction on range. The default value is 1.0.

A theme's weight in effect represents the cost of translating that theme relative to the translation cost of the other themes in the same source dataset. For example, a contour line theme having many features and/or vertices would usually have a higher weight value than a political boundary theme containing fewer features and/or vertices.

Specify Query Coordinate System

Each DDE translation request specifies a spatial extent from which features should be obtained. The bounds of this extent define the query rectangle used by DDE to clip features to the desired area. These bounds values are entered into the Search Area fields of the default Order Form page, and are also specified in remote fetch URLs.

The coordinate system in which the query rectangle bounds are expressed can be the same as or different from the coordinate system of the source data. When source data originates from a single source, the query coordinate system is typically the same as the source coordinate system. When source data originates from multiple sources, the query coordinate system is either the same as one of the source coordinate systems, or different from all of them.

In any case, the coordinate system of the query rectangle bounds must be made known to DDE. This is done by modifying a setting in the following file:

```
<DDEInstallDir>/translationControl/system/system.fmi
```

The `system.fmi` file contains a line defining a macro representing the query coordinate system. As shipped, the default macro specifies the LL84 coordinate system, which is correct for the supplied sample data. The default macro line is:

```
DEFAULT_MACRO QueryCoordSys LL84
```

This line must be modified to specify the query coordinate system to be used with the user data. For example, if query extent bounds will be expressed in the Bipolar Oblique Conformal Conic coordinate system, the line would be changed to:

```
DEFAULT_MACRO QueryCoordSys BPCNC
```

For source datasets having no defined coordinate system, the line should specify a query coordinate system value of `UNDEFINED`.

Where to Find the Coordinate System Names

The specified coordinate system value must be one of the coordinate system names defined and used by the FME. For example, the name LL84 represents Lat/Long with the WGS84 datum. These names can be found as follows:

On Windows Systems

Start the FME Universal Translator in its interactive GUI mode, and select Tools | Browse Coordinate Systems. The displayed Coordinate System Gallery table will list all defined FME coordinate systems. Any name in the Name column is a valid value for the query coordinate system macro.

On UNIX Systems

Obtain the complete list of FME coordinate system names on UNIX systems by examining the contents of the `<FMEInstallDir>/coordsys.db` text file. Each row, which is separated into pipe-delimited fields, represents an FME coordinate system. Any name in the first field is a valid value for the query coordinate system macro. The second field contains a description of the coordinate system for easier identification.

Specify Source Data Spatial Extent

The spatial extent of the source data must be specified for DDE. This involves modifying Translation Servlet property values. Where these values are stored depends on the servlet engine that is used to run the Translation Servlet.

User-Provided Servlet Engine

If a user-provided servlet engine/web server is being used to run the Translation Servlet, the property values must be modified in a manner specific to the servlet engine/web server in use. Please refer to your particular servlet engine/web server's documentation for details on how to modify servlet property values.

DDE-Provided Servlet Engine

If one of the default bundled DDE servlet engines (Tomcat or ServletRunner) is being used to run the Translation Servlet, the property values are modified in one of two possible files, depending on which operating system DDE is being run on. These are described as follows:

On Windows and Linux Systems

On Windows and Linux systems, the bundled Tomcat servlet engine is used. To specify the source data spatial extent on these systems, edit the following file:

```
<DDEInstallDir>/tomcat/webapps/servlet/WEB-INF/web.xml
```

and change the values assigned to the following four properties:

```
minXSearchLimit
maxXSearchLimit
minYSearchLimit
maxYSearchLimit
```

The `web.xml` file specifies properties that determine the behaviour of the Translation Servlet. It contains many properties in addition to the four listed above. The property names and their values are specified using the following XML tag syntax (the parameter names and sample values for each are shown in boldface text within the tags):

```
<init-param> <param-name>minXSearchLimit</param-name>
<param-value>-140.0</param-value> </init-param>

<init-param> <param-name>maxXSearchLimit</param-name>
<param-value>-109.0</param-value> </init-param>

<init-param> <param-name>minYSearchLimit</param-name>
<param-value>48.0</param-value> </init-param>

<init-param> <param-name>maxYSearchLimit</param-name>
<param-value>62.0</param-value> </init-param>
```

You may have to scan through the file to locate these four properties. Modify only the values of these properties; leave all other values as-is.

The properties should be set to values corresponding to the source data's min/ max x and y coordinates.

For example, the DDE sample source data is in Lat/Long decimal degrees, with an extent bounded by Latitudes 48N-62N and Longitudes 140W-109W. The corresponding properties would be assigned values of:

`minXSearchLimit`	`-140.0`
`maxXSearchLimit`	`-109.0`
`minYSearchLimit`	`48.0`
`maxYSearchLimit`	`62.0`

Note the use of negative values to indicate western longitudes.

Note: The coordinate values must be in the same coordinate system defined for the query rectangle as described in *Specify Query Coordinate System* on page 117.

On All Other Systems

On systems *other* than Windows and Linux, the bundled ServletRunner servlet engine is used. To specify the source data spatial extent on these systems, edit the following file:

```
<DDEInstallDir>/Jsdk/webpages/WEB-INF/servlets.properties
```

and change the values assigned to the following four properties:

```
minXSearchLimit
maxXSearchLimit
minYSearchLimit
maxYSearchLimit
```

The `servlets.properties` file specifies properties that determine the behaviour of the Translation Servlet. It contains many properties in addition to the four listed above. You may have to scan through the file to locate these four properties. Modify only the values of these properties; leave all other values as-is.

These properties should be set to values corresponding to the source data's min/ max x and y coordinates.

For example, the DDE sample source data is in Lat/Long decimal degrees, with an extent bounded by Latitudes 48N-62N and Longitudes 140W-109W. The corresponding properties would be assigned values of:

`minXSearchLimit`	`-140.0`
`maxXSearchLimit`	`-109.0`
`minYSearchLimit`	`48.0`
`maxYSearchLimit`	`62.0`

Note the use of negative values to indicate western longitudes. Also note the "\" backslash character at the end of some of the lines in the file. This is used as a line continuation character.

Note: The coordinate values must be in the same coordinate system defined for the query rectangle as described in *Specify Query Coordinate System* on page 117.

Specify Source Data Formats

Specifying the source data format(s) involves associating the reader keyword(s) with their format(s) and then creating the importer file(s) that will define the characteristics of the source data.

Associate Reader Keywords with Source Formats

To specify the data source format(s) that the source data is stored in edit the following file:

```
<DDEInstallDir>\translationControl\importers\multiSource.fmi
```

and find the line near the beginning of the file that starts with:

```
MACRO readerKeyword2Type
```

The value of this `readerKeyword2Type` macro needs to be changed to reflect the data format(s) associated with the reader keyword(s) chosen to represent the data source(s). As shipped, the default definition (reflecting the single sample data source) is:

```
   MACRO readerKeyword2Type sample FFS
```

This definition states that the `sample` data source is stored in FFS format.

The macro value's syntax consists of one or more `<readerKeyword>` `<readerType>` pairs, listing each data source's keyword and its associated format type, separated by spaces. The format is:

```
MACRO readerKeyword2Type <readerKeyword_1> <readerType_1>
... <readerKeyword_n> <readerType_n>
```

Each `<readerKeyword_n>` represents a data source keyword. The keywords used here must be the same ones used as suffixes appended to theme names in `config.csv` as described in *Data Source Keywords* on page 115.

Each `<readerType_n>` must be a valid FME Reader type, such as `SHAPE` or `SDE30`.

For example:

```
MACRO readerKeyword2Type municipal SHAPE county SHAPE
provincial SDE30 federal SHAPE commercial SDE30
industrial SDE30
```

The municipal, county and federal sources are in `SHAPE` format, while the provincial, commercial and industrial sources are in `SDE30` format.

In practice, most DDE installations will typically use only one or perhaps two data sources, and will therefore need to define only one or two data source keywords.

Create Importer Mapping Files

An importer mapping file must be created for each data source. This is accomplished by making copies of supplied, format-specific importer template files and modifying each copy to create a site-specific importer for each separate data source.

The importer template files have names whose syntax is `<sourceFormat>template.fmi`. These are located in:

```
<DDEInstallDir>\translationControl\importers\multiSource\
user
```

All site-specific importers made from these templates must also be stored in this location.

The currently available set of importer file templates is listed in Table 11-1 .

Note: Importer templates for alternative source data formats not listed in Table 11-1 can be created on request. Contact your local ESRI representative for further details.

TABLE 11-1 Importer Mapping Files

Mapping File	Supported Format
`GEODATABASE_MDBtemplate.fmi`	ESRI GeoDatabase (Access) (*Windows only*) 1
`GEODATABASE_SDEtemplate.fmi`	ESRI GeoDatabase (SDE) 1

Mapping File	Supported Format
`SDE30template.fmi`	ESRI Spatial Database Engine 3.x/ ArcSDE 8.x
`SHAPEtemplate.fmi`	ESRI Shape

1. Requires ESRI ArcGIS.

To create the required importer files, perform the following steps for each data source:

1 Create a copy of the template file associated with the format of the data source.

2 Rename the copy so that its filename is the keyword of the data source. The keyword must be present in the `readerKeyword2Type` macro value described in *Specify Source Data Formats* on page 121.

For example, if two `SDE30` data sources and one `SHAPE` data source are used, and they are accessed using the source keywords "municipal", “provincial” and "federal" respectively, then two copies of `SDE30template.fmi` and one copy of `SHAPEtemplate.fmi` would be made and renamed as follows:

```
SDE30template.fmi (copy 1) --> municipal.fmi
SDE30template.fmi (copy 2) --> provincial.fmi
SHAPEtemplate.fmi (copy 1) --> federal.fmi
```

Define Source Data

Once the required importer template copies have been made, the location, coordinate system and other details of the source data must be defined for DDE.

To do this, edit the site-specific macro definitions within each renamed template copy created in the previous step, substituting appropriate values for the placeholders enclosed in angle brackets (< >). Each copy includes a comment section at the beginning describing the macro values to modify and examples for each. If in doubt, please see the FME Readers and Writers manual for further details concerning specific format parameters.

The macros requiring modification vary according to the source format being imported. However, all importers include the following two macros:

1 **SourceCoordSystem**

This macro represents the FME keyword for the coordinate system the source data is stored in. Example values are LL (Latitude/Longitude), UTM-12N (UTM Zone 12 North) and TX-S (Texas State Plane NAD27 South).

2 **ReaderKeyword**

This macro represents the reader keyword associated with the importer file. The value of this macro must be the same as the file's name.

In addition, all non-database importers include the following macro:

- SourceDataset

  This macro represents the directory path or file path to the source data. If in doubt, please see the FME Readers and Writers manual for details on whether a format is directory or file based.

Example

Using the example in the previous section, the `federal.fmi` importer file (specifying SHAPE format) would be modified such that its `SourceDataset`, `SourceCoordSystem` and `ReaderKeyword` macros are assigned the desired site-specific values (sample values are shown here):

```
MACRO SourceDataset d:\tmp\shape_directory
MACRO SourceCoordSystem LL84
MACRO ReaderKeyword federal
```

The `provincial.fmi` importer file (specifying SDE30 format) would have the following macros set to the desired site-specific values (sample values are shown here):

```
MACRO SourceCoordSystem BCALB-83
MACRO SDE30Dataset sde
MACRO SDE30Server capricorn
MACRO SDE30UserID bill
MACRO SDE30Password jfdjeq
MACRO SDE30Instance sde1
MACRO ReaderKeyword provincial
```

Undefined Source Coordinate System

For source datasets having no defined coordinate system, set the value of the `SourceCoordSystem` macro to be `UNDEFINED`.

Restart DDE

Once all of the preceding steps have been completed, DDE can be restarted.

CHAPTER 12

FME Translation Server

The FME Translation Server (*FME Server*) is the fundamental component of DDE and exploits all of the capabilities of FME translation engine. The FME Translation Server is responsible for performing translation requests and making the results available to the calling application.

Unlike the standard version of FME, when the FME Server is started, it goes into server mode where it awaits translation requests. All interaction between the FME Server and external applications takes place via the FME Server API. This API enables applications written in C/C++ or Java to send translation requests directly to the FME Server.

This section discusses how the FME Server operates, how it's installed, configured and started, and how third-party applications interface with it to exploit its capabilities. In addition, it will briefly discuss the set of mapping files used by the FME Server.

Theory of Operation

The FME Server is a web-enabled version of FME. It contains all of the functionality of the FME along with the ability to listen on a TCP/IP port for remote translation requests. The architecture of the FME Server and the external software components it is dependent upon are shown in Figure 4-1.

The FME Server is simply the FME running in its server mode. This mode allows it to run in one of two ways. It either:

- starts up and listens on a port for clients to come to it and give it translation requests.
- starts up and connects to a port on which a client is already listening, waiting to send translation requests. This is the variant used in DDE.

FME can be run in server mode on its own, independent of the rest of the DDE components, as long as its server mode is activated by the appropriate license keycode. Of course, in order to be useful the FME needs a client of some sort to send it requests when running in server mode. In the default DDE environment this client function is supplied by the QServer.

A translation request is submitted by an external client application (such as the QServer). The request consists of an FME command string as it would be typed on a command line to cause FME to run a mapping file and perform a translation. The FME Server takes this request, executes it and upon completion returns a translation status string to the client application. In a DDE environment, if the translation is successful, the status string includes a URL pointing to the translation result.

The FME Server registers itself with an external application client such as the QServer. On startup the FME Server sends a registration request to the client, which returns a registration confirmation, informing the FME Server of the port on which to listen for subsequent translation requests.

FME Server configuration is specified by a configuration file. On startup the contents of this file are read by the FME Server and used to set up its operating environment.

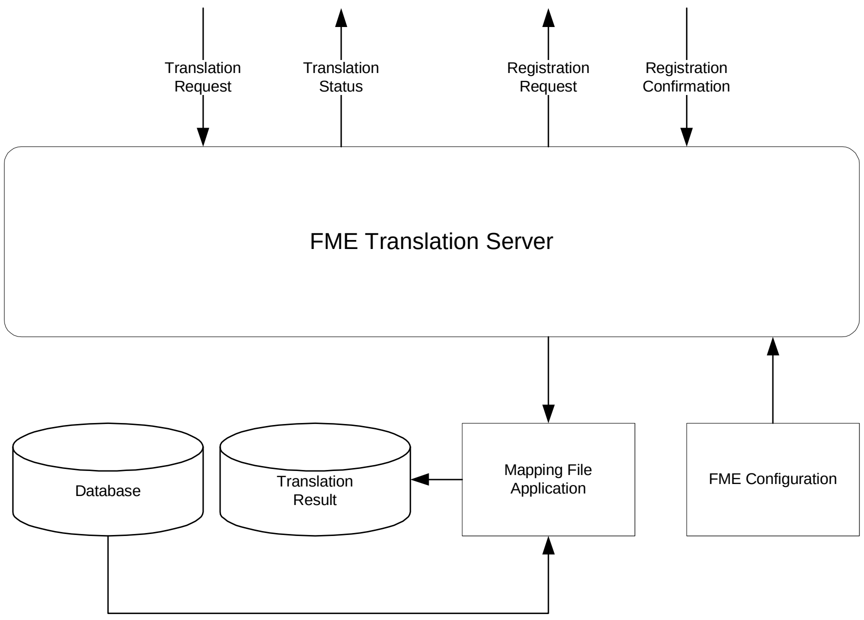


FIGURE 12-1 **FME Translation Server**

FME Server Installation

The FME Server is installed automatically as part of the DDE installation procedure. The installed FME Server is located in:

```
<DDEInstallDir>/fme
```

FME Server Configuration

Three configuration considerations exist to ensure the FME Server will run correctly for a given environment: the FME Server configuration file, the FME mapping files, and the FME Server's file access. When DDE is initially installed, it is configured to run with a set of default values. It is suggested that DDE be installed with all of its defaults, then tested using the supplied sample source data, and then reconfigured for the site-specific environment.

FME Server Configuration File

An optional configuration file can be specified when the FME Server is started. This configuration file contains directives that instruct the FME Server on how to initially start, what to do prior to the execution of a request, what to do after the execution of a request, and the contents of its return status messages.

When running within the DDE environment, the FME Server uses a configuration file created specifically for DDE operations.

The configuration file is located in the root installation directory of DDE. It is an ASCII text file which is structured into several sections, each of which performs a specific set of tasks. The general form of the configuration file is described in *Interfacing with the FME Server* on page 130.

When changes are made to the configuration file, the FME Server must be halted and restarted for the changes to come into effect.

FME Mapping Files Used in DDE

The DDE translations carried out by the FME Server are controlled by a default set of FME mapping files located in the `translationControl` subdirectory tree within the DDE installation directory. These mapping files include format-specific importers and exporters, as well as common utility INCLUDE files and Tcl scripts.

FME Server File Access

The set of FME mapping files and the data source must exist on a file system to which the FME Server has read and write permissions. In addition, the directory where the translation results and log files are written must also allow read and write access to the FME Server.

From the client's perspective, if it doesn't share the same file system with the FME Server, then the client will require sufficient information to allow it to access the results generated by the FME Server. Such access could be through a URL provided by the FME Server or via pre and/or post commands as specified in the FME Server configuration file. The latter allows for relocating files to a visible file system before and/or after a translation.

FME Server Start-up and Shutdown

The FME Server is usually started automatically by the Process Monitor as part of the latter's sequenced start-up of all DDE components. This is the usual and recommended start-up procedure and is described in Chapter 2, *Getting Started*. However, the FME Server can also be started manually if desired.

When used with DDE, the FME Server should be run from an account or process which does not have Administrator or root-level privileges. This is a safety precaution because the FME Server (when used with DDE) periodically deletes all files in the translation results directory older than a certain age. If this is done with high-level privileges, *and* if erroneous settings were made during installation, the FME Server may possibly delete many other files unintentionally.

Manual Start-up

The FME Server can be manually started on the command line. The command takes one of two forms:

```
FME REGISTER_SOCKET <host> <ClientserviceName|ClientPort>
[<config >]
FME CREATE_SOCKET <serviceName | port> [<config>]
```

In the first case, the FME Server connects to a client (such as the QServer) by registering itself with the client that is running on `<host>` and listening on `<Clientport>`. This is the form of FME Server start-up used for DDE operations.

In the second case, the FME Server creates a socket and waits for remote translation requests on a specific port (a client application must have prior knowledge of the port on which the FME Server will be listening).

The following table describes the command-line arguments to the FME Server.

Argument	Description
`serviceName`	The name of an entry in the operating system's `services` file. The entry corresponds to a port number that is used by the server.
`port`	The port number the FME Server will use to connect to clients.
`config`	The name of the FME Server configuration file that defines the keywords and associated operations to be performed before and after each translation.
`host`	The host name of the machine on which the client application is running.
`ClientserviceName`	The name of an entry in the `services` file of the client host. The entry corresponds to the port number that is used by the client.
`Clientport`	The port number the client will use for connections.

If a service name rather than an explicit port number is to be used, the operating system's service file should be edited to define the service name and associate it with a port number. On UNIX, the services file is usually located in `/etc/services`. On Windows, it is usually in `\winnt\system32\drivers\etc\services.`

Automatic Start-up as a Service

There will be cases when an enterprise will require the FME Server to stop and start automatically, independent of a user account, when the system it resides on is rebooted. To achieve this on Windows, you must add the start-up of the FME Server as a service in the Control Panel | Services menu. Appendix B, *DDE as a Windows Service* describes how to install DDE and its components as services on Windows assuming the existence of the Process Monitor. Automatic shutdown and start-up can be implemented in UNIX by adding an entry to the `/etc/init.d` file.

Shutdown

It is recommended that the FME Server be shut down as part of the overall DDE shutdown procedure. Please see Chapter 2 for details on DDE shutdown.

If the FME Server was started indirectly by a procedure (for example, as a Windows Service), then the shutdown mechanism provided by that procedure should be used to shut down the FME Server.

FME Server Logging

During its operation, the FME Server does not generate a specific log file of its server activities. Rather, each translation it performs generates a log file that contains information about the translation. The location of these log files is set in the FME Server configuration file.

Interfacing with the FME Server

DDE is an integrated application. However, each component, including the FME Server, can interface with third-party applications. Since the FME Server represents the bottom of the DDE hierarchy, the third-party application takes the role of client.

FME Server API

DDE currently offers three client APIs for the FME Server: C++, C, and Java. Each of these APIs furnishes method calls that return a value of 0 for success and a non-zero integer for failure.

In C++, the client API is provided by a single class named `FMESocketClient()`.

In C, the client API is provided through a set of function calls. These calls closely mimic the behavior of the C++ classes.

In Java, the client API is provided through two classes named `FMESocketClient()` and `FMESocketRegisterClient`. The first `FMESocketClient()` provides the functionality for a client to connect to a standalone FME Server and the class `FMESocketRegisterClient` provides the functionality when the FME Server is to register with the client.

FME Server Configuration File

An FME Server configuration file can contain three components: a server name, a global section and zero or more subsections. Their general layout is:

```
FME_SERVER_NAME <name>

GLOBAL_SECTION

     <Global Directives>

SUB_SECTION <subsection_name>

     <Subsection Directives>
```

The FME Server configuration file present in the default DDE distribution provides an example of the information discussed in the following sections. This configuration file is located in:

```
<DDEInstallDir>\fmeServerConfig.txt
```

The following sections describe the configuration file components.

FME_INSTANCE_NAME

This is an optional directive that specifies the name of the FME Server instance. This is most useful when there are multiple FME Servers connected to the QServer. The QServer's Selective Load Distribution (SLD) mechanism uses the FME instance name defined here to distinguish between multiple FME Servers.

If this value is not supplied in the configuration file, then a default value of `fme` is used. The directive has the following form:

```
FME_INSTANCE_NAME <serverName>
```

GLOBAL_SECTION

The `GLOBAL_SECTION` defines all of the global directives that apply to a single server session. A server session is defined as the period of time when an FME Server is started up, performs some number of translations, and is shut down. The possible directives are described in the table below.

Directive	Description	Default
`FME_WORKING_DIR <translation-working-directory>`	Specifies the path of the directory into which the FME Server will write all of its translation results. See the warning below.	Current Working Directory
`FME_MAPPING_DIR <translation-mapping-file-directory>`	specifies the path of the directory containing the mapping files used by the FME server during translations	Current Working Directory
`FME_RESULT_LIFETIME <translation-result-lifetime-seconds>`	Specifies how long translation results will be held for in seconds. FME will periodically check for and delete translation result files older than this amount from the `FME_WORKING_DIR` as a housekeeping measure. The frequency of this automatic deletion check is set by the value of `FME_PURGE_INTERVAL`, described below. Whenever automatic file deletion is performed, it occurs after a translation.	If directive is absent, results are never deleted – lifetime is infinite.

Directive	Description	Default
`FME_PURGE_INTERVAL <translation-result-deletion-interval-seconds>`	Specifies the period for automatic deletion checking in seconds. FME will perform a check for files older than `FME_RESULT_LIFETIME` and delete any it finds. This check is performed at the `FME_PURGE_INTERVAL` period. Note that the FME decides whether or not to do the check only after each translation. It does not set a timer to trigger the check. This means that the check can occur at the interval set, but may also occur at longer intervals depending on how frequently translations are made. A value of zero means that the check is performed after every translation.	If directive is absent, the default is 0 – will check after each translation if there are translation results to purge
`SDE30_PERM_CONNECT <host> <instance> <database> <userID> <password>`	Defines a permanent connection to an SDE Server. This connection is brought up just before the first translation is performed on the FME Server, and is then held by the FME Server so subsequent translations need not establish a new SDE connection. Take care when using this directive since SDE connections are a valuable resource and should be used sparingly. In general, it is good practice to first use the FME Server without any permanent connection. You might consider a permanent connection later, if you find that connecting and disconnecting to SDE is too expensive. For values, see *Values for the SDE Connection String* below.	No default supplied as it is site-specific.

TABLE 12-1 Values for the SDE Connection String

Argument	Description	Default
`<host>`	The name of the host computer on which the SDE server is running.	site-specific
`<instance>`	The SDE instance to which the FME Server is to connect.	esri_sde
`<database>`	The database on the instance which is to be connected. When the SDE is on databases such as Oracle the value specified is not used. Although any value can be specified the convention is to simply specify the value `NOTUSED` for the database.	site-specific
`<userID>`	The user account used to log in to the SDE	Site-Specific
`<password>`	The user password of the user account.	Site-Specific

Warning

Take care to ensure that the `FME_WORKING_DIR` value is set to the appropriate directory into which the FME Server will write its translation results.

This is important because the FME Server periodically and recursively deletes all files in this directory that are older than a certain age (specified by `FME_RESULT_LIFETIME`). If the `FME_WORKING_DIR` is set to the wrong directory, the FME Server will be deleting files in this directory instead.

As an extreme example, if `FME_WORKING_DIR` was erroneously set to the UNIX "/" `root` directory, the FME Server would periodically and recursively delete all files in the UNIX file system if it was running as `root` (it is recommended that the FME Server be run from a non-privileged account or process).

SUB_SECTION

The `SUB_SECTION` is optional. If a subsection is present, it is named with a keyword by which it can be referenced by client applications. Multiple subsections can be defined for various purposes.

Each subsection can specify operations the FME Server should perform before and after a translation request has been processed. The subsection can also define what translation success and failure response messages should be returned to the client, and define FME mapping file macro values as well.

By specifying a subsection by name in the translation request, clients can cause the operations defined by the subsection to be performed by the FME Server for the translation being requested.

Each subsection has the following general form:

```
SUB_SECTION <keyword> \
[MACRO_DEF <macroName> <macroValue> \]*
[PRE_COMMAND <pre-command> \]*
[POST_COMMAND <post-command> \]*
[SUCCESS_RESPONSE <message>  \ ]
```

```
      [FAILURE_RESPONSE <message>  \ ]
```

Directive	Description
`<keyword>`	Identifies the subsection being defined. There is no limit to the number of subsections that can be defined. Each subsection must be identified with a unique `<keyword>` and cannot begin with `FME_`
`MACRO_DEF <macroName> <macroValue>`	This defines a macro that is to supplied to the mapping file when the translation is performed. The first parameter `<macroName>` specifies the name of the macro. All the tokens up to the next Directive or the end of the `SUB_SECTION` definition form the value for `<macroValue>.`
`PRE_COMMAND <pre-command>`	This specifies a command that is performed by the server before a translation. There is no limit to the number of `PRE_COMMANDS` that can be defined in a `SUB_SECTION`. The commands are executed in the order they are specified in the `SUB_SECTION`. Any command that can be executed from a command line from an operating systems window can be specified as a `PRE_COMMAND`.
`POST_COMMAND <post-command>`	This specifies a command that is performed by the server after a translation. There is no limit to the number of `POST_COMMANDS` that can be defined in a `SUB_SECTION`. The commands are executed in the order they are specified in the `SUB_SECTION`. Any command that can be executed from a command prompt can be specified as a `POST_COMMAND`.
`SUCCESS_RESPONSE <message>`	The format of the message to return to the client when the translation is successful.
`FAILURE_RESPONSE <message>`	The format of the message to return to the client when the translation fails.

Response Messages

The `SUCCESS_RESPONSE` and `FAILURE_RESPONSE` directives described in the table above define the message string that will be returned by the FME Server for successful and failed translations respectively. These directives provide the mechanism by which the FME Server communicates results back to the client.

The content of these response messages is meaningful only to the client. The FME Server simply passes the message as defined back to the client, which then processes it in whatever manner it chooses. The message can include the predefined pseudo-variables (described below) as well as several other directives.

In a DDE environment, the following response messages are defined:

Success Response Message

The success response message consists of a number of fields delimited by "|" pipe characters. Most fields have the form `<parameterName>=<parameterValue>`. The syntax is as follows:

```
SUCCESS_RESPONSE 0:TranslationSuccessful|ResultPre-
fix=<resultPrefix>|ResultRootDir=<resultRootDir>|
ResultDataset=!FME_AUTO_FILE_NAME.gif!|Result-
Log=!FME_AUTO_FILE_NAME.log!|NumFeaturesOut-
put=!FME_NUM_FEATURES_OUTPUT!|
ResultLifetime=!FME_RESULT_LIFETIME!
```

The individual fields are as follows:

`0:Translation Successful`

The general translation success status string.

`ResultPrefix=<resultPrefix>`

The web communication protocol to use. `<resultPrefix>` can be any valid web protocol that is available and allows translation results to be accessed. It represents the URL prefix that the Translation Servlet will use to access the FME server's translation results. Examples are http: and ftp:. The trailing colon is a required part of the value.

`ResultRootDir=<resultRootDir>`

The URL pathname of the web root directory containing the directory into which the FME server places all of its translation results. `<resultRootDir>` must include a pair of leading forward slashes. An example is //jim/results.

```
ResultDataset=!FME_AUTO_FILE_NAME.gif!
ResultDataset=!FME_AUTO_FILE_NAME.zip!
```

The FME server-local pathname of the translation result, either for displayable GIF output or non-displayable zip-compressed output. The `FME_AUTO_FILE_NAME` pseudo-variable is used to provide a unique temporary file name.

`ResultLog=!FME_AUTO_FILE_NAME.log!`

The FME server-local pathname of the translation result log file. The `FME_AUTO_FILE_NAME` pseudo-variable is used to provide the same unique temporary name for the log file that is used for the associated result.

`NumFeaturesOutput=!FME_NUM_FEATURES_OUTPUT!`

The number of output features in the translation results. The `FME_NUM_FEATURES_OUTPUT` keyword is not a pseudo-variable but rather a placeholder for the actual number of output features.

`ResultLifetime=!FME_RESULT_LIFETIME!`

The retention lifetime of the translation results in seconds. The value here is the same as the value specified by the `FME_RESULT_LIFETIME` directive in the `GLOBAL_SECTION`.

Failure Response Message

The failure response message consists of a number of fields delimited by “|” pipe characters. Most fields have the form

```
<parameterName>=<parameterValue>
```

The syntax is as follows:

```
FAILURE_RESPONSE !FME_ERROR_NUMBER!:!FME_ERROR_MSG!|
ResultPrefix=<resultPrefix>|ResultRootDir=<resultRootDir>|
ResultLog=!FME_AUTO_FILE_NAME.log!
```

The individual fields are as follows:

`!FME_ERROR_NUMBER!:!FME_ERROR_MSG!`

The translation failure status string. The pseudo-variables `FME_ERROR_NUMBER` and `FME_ERROR_MSG` are used to assign the specific FME translation error information.

The remaining fields are identical to the same-named ones used in the success response message.

Pseudo Variables

Within the `GLOBAL_SECTION` and `SUB_SECTION` definitions there are a number of *pseudo-variables* that can be specified if desired. Pseudo-variables act as placeholders which are replaced at translation time with the appropriate values as described in the table below. Pseudo-variable names are always enclosed within exclamation marks. Pseudo-variables can be used for customizing the behaviour of the FME Server to aid in file and directory creation and to incorporate specific information in the response messages.

The pseudo-variables that can be specified are as follows:

Pseudo-Variable Name	Replacement Value Description
`!FME_AUTO_FILE_NAME!`	An auto generated file name that is guaranteed to be unique.[1] The file name is generated before the translation so that multiple references to `!FME_AUTO_FILE_NAME!` will identify the same file throughout a single translation. The file name location is relative to the path specified by the `FME_WORKING_DIR` global directive.
`!FME_AUTO_DIR_NAME!`	An auto generated directory name that is guaranteed to be unique. The directory name is generated before the translation so that multiple references to `!FME_AUTO_DIRECTORY_NAME!` will identify the same directory throughout a single translation. The directory name location is relative to the path specified by the `FME_WORKING_DIR` global directive.
`!FME_ERROR_MSG!`	The contents of the error message that contains the reason the translation failed. This value is not available to the `PRE_COMMANDS.`
`!FME_ERROR_NUMBER!`	The FME internal error number associated with the error message that is returned.
`!FME_NUM_FEATURES_OUTPUT!`	The number of features in the translation output.
`!FME_RESULT_LIFETIME!`	The value assigned to the `FME_RESULT_LIFETIME` global directive.
`!FME_SERVER_NAME!`	The value assigned to the `FME_SERVER_NAME` directive if present.
`!`*`fmeMacroName`*`!`	The value assigned to the named FME macro. The value of any macro defined to the FME during translation can be accessed by a pseudo-variable whose name is the same as the macro name.

1. If a filename extension is included with this pseudo-variable, different behaviour will result depending on whether the extension is included on the inside or the outside of the exclamation point delimiters of the `!FME_AUTO_FILE_NAME!` pseudo-variable. The default FME Server configuration uses extensions that are inside the exclamation points, such as:

```
!FME_AUTO_FILE_NAME.log!
!FME_AUTO_FILE_NAME.zip!
```

Extensions inside the exclamation points cause *different* filenames to be generated for each occurrence of the same pseudo-variable name. This is by design, in order to guarantee that multiple occurrences of `!FME_AUTO_FILE_NAME.<whatever>!` (if they existed) produce unique, non-conflicting filenames.

Extensions outside the exclamation points cause the *same* filename to be generated for each occurrence of the same pseudo-variable name. So for example, this:

```
!FME_AUTO_FILE_NAME!.log
!FME_AUTO_FILE_NAME!.zip
```

would give the same name for both the .log and the .zip file.

Determination of Result Location

In a DDE environment, when the FME Server sends a response message back to the Translation Servlet, the latter constructs a complete URL for the result location by concatenating the value of the `ResultPrefix` and the value of the `ResultRootDir` (both contained in the response message) and appending to this the translation results filename that was auto-generated by the FME Server. The complete URL is then sent by the Translation Servlet to the client, which uses it to access the results.

For example, if the following FME Server settings were in effect:

```
FME_WORKING_DIR /translations/results
ResultPrefix=http:
ResultRootDir=//WORF/translations/results
```

and the auto-generated result file was `dog.zip,` the Translation Servlet would construct the following result URL and send it to the client:

```
http://WORF/translations/results/dog.zip
```

Using this URL, the client (typically a browser) would then access the file `dog.zip` located in `translations/results` through the http web server running on host WORF.

CHAPTER 13

QServer

The QServer is an intermediate layer in DDE that receives translation requests from client applications, such as the Translation Servlet, and hands those requests to the first available FME Server. If all FME Servers are currently busy performing translations, the QServer places the request into a queue where it waits until an FME Server becomes available. In addition, the QServer can accept client-specified priorities and delayed processing times on a per-request basis.

Upon completion of the translation request by an FME Server, the QServer receives a status message from the latter and sends it back to the client application. The returned status message typically includes the location of the actual translation results as determined by the FME Server, thereby enabling the client to access them.

The QServer can automatically restart an FME Server, based on the number of failed and successful translations performed by that Server. The QServer also maintains an extensive log file of its operations.

Theory of Operation

Client interaction with the QServer is performed through the QServer API which provides network TCP/IP communication between the QServer and the client applications. Figure 13-1 shows the architecture of the QServer.

.

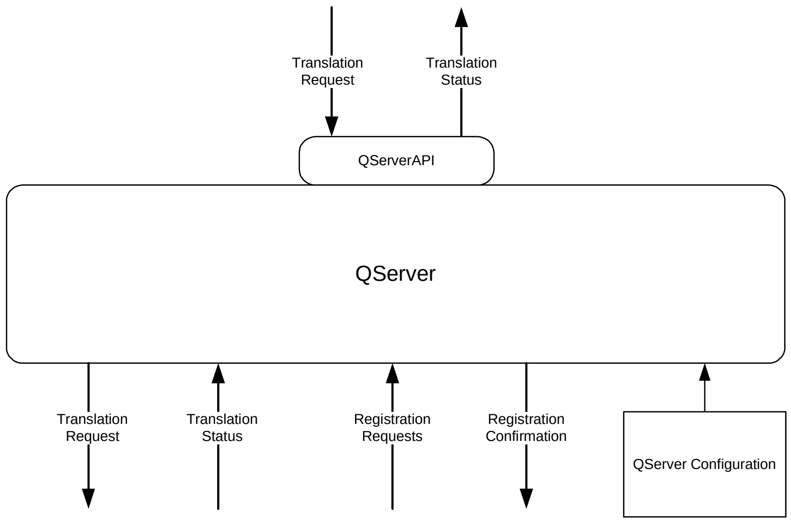


FIGURE 13-1 **QServer Architecture**

The flow of data into and out of the QServer is shown in Figure 13-2.

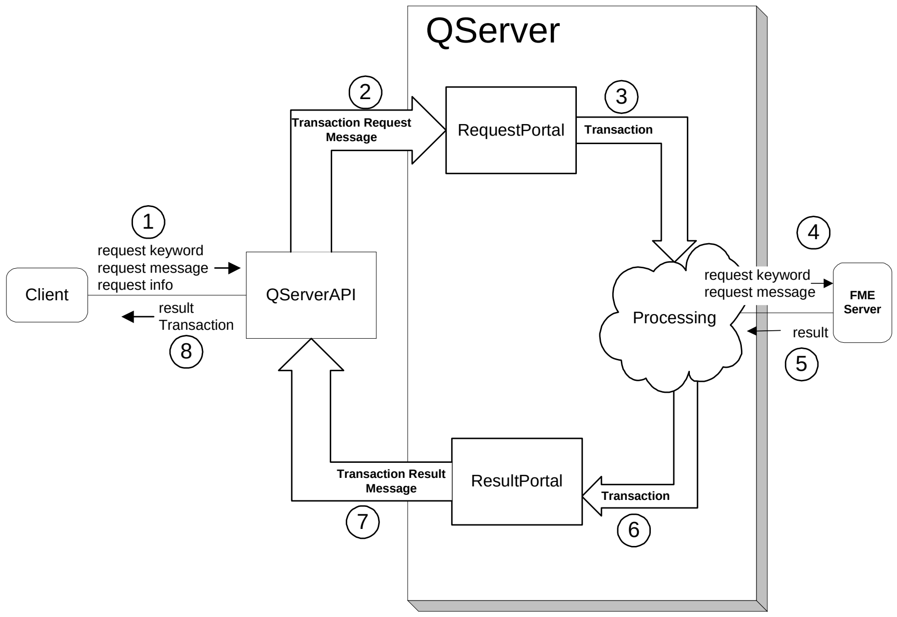


FIGURE 13-2 **QServer Data Flow**

The QServer listens for client translation requests on a specific port, accepting and forwarding them on to waiting FME Servers. If all available servers are busy performing requested translations, the QServer will place subsequent incoming requests into a queue, sending the request at the head of the queue to the next FME Server that becomes available. Translation requests can also be placed onto the queue according to a client-specified priority level.

A translation request can optionally specify a time in the future when the request will be processed by the QServer. The request is held by the QServer until that specified time, after which it is placed in the queue for processing.

FME Servers register themselves with the QServer when they are first started. The QServer listens for these registration requests on a specific port. Once an FME Server has registered itself with the QServer, the latter sets up a TCP/IP communication link with the FME Server that remains open until the FME Server is shut down. A separate link is established for every registered FME Server. Subsequent translation requests are sent by the QServer to each FME Server over its associated link.

Requests to an FME Server may fail and, in certain cases, the QServer may restart the request. There are two types of aborted translation handled by the QServer:

- **FME Server Unavailable**: In this type of failure, the QServer attempts to send a translation request to an FME Server which is no longer available to accept the request. In this case, the QServer re-queues the request by placing it back into the request queue.

  For this type of failure, error handling is performed entirely within the QServer. The problem is never reported to the client, although the QServer's logfile indicates that the error occurred.
- **Returned Status Unavailable:** In this second type of failure, the QServer successfully sends a translation request to an FME Server, but the latter fails to return a result. In this case, the QServer returns a result via the QServer API with a service failure indication. The QServer API detects this failure indication and throws an exception that must be caught and processed by the client in whatever manner it deems necessary.

Request Priority Support

The QServer supports a simple request priority mechanism. Priority can be programmatically specified as a parameter through the QServer API, or as a parameter within a remote fetch URL.

Priority values must be integers >= 0 with higher values meaning higher priority. There is no logical restriction on the maximum value. If a request's priority value is < 0 or is not specified, the QServer sets it to 1.

Requests with higher priority are inserted into the request queue ahead of requests with lower priority. If requests with the same priority as a new request are already present, the new request is inserted after the last of those requests.

QServer Selective Load Distribution (SLD)

The QServer implements a form of load balancing known as selective load distribution, or SLD. SLD is used with DDE systems running more than one concurrent FME Server, and allows the QServer to distribute translation requests to specific FME Servers based on a number of user-configurable acceptance conditions.

Translation requests that do not meet any of the specified acceptance conditions remain in the QServer request queue until either an FME Server registers itself that has conditions that do accept the request, or the QServer is shut down (losing the request).

SLD behavior is optional in DDE systems and can be enabled or disabled as desired. Enabling SLD is useful only when two or more FME Servers are in concurrent use. Enabling it for a single FME Server is also valid, but in most environments doing so will not provide any additional benefits.

The parameter that enables/disables SLD is located in the QServer's configuration file, by default named `<DDEInstallDir>/qServerConfig.txt`. The parameter takes a value of either `"true"` or `"false"` and has the following syntax:

```
ENABLE_SELECTIVE_LOAD_DISTRIBUTION=false
```

By default, SLD is initially disabled in the standard DDE distribution.

SLD setup involves naming the FME Servers and specifying the acceptance conditions under which each will accept and perform translations. For modifications made to SLD parameters to take effect, DDE must be restarted.

FME Server Naming

In a multiple FME Server environment, individual FME Servers are identified to the SLD mechanism by unique FME instance names. These names are user-defined and are specified in each FME Server's configuration file (by convention, named `<DDEInstallDir>/fmeServerConfig.txt`) using the `FME_INSTANCE_NAME` directive as the first line in the file. The syntax is:

```
FME_INSTANCE_NAME <fmeServerName>
```

The `<fmeServerName>` is user-defined and should contain only letters, digits and underscores. For example,

```
   FME_INSTANCE_NAME Green
```

or

```
   FME_INSTANCE_NAME fastServer_2
```

If the `FME_INSTANCE_NAME` directive is absent, the default name of an FME Server instance is `"fme"`.

For multiple FME Server environments to take advantage of SLD, each FME Server must have a unique instance name as specified by the `FME_INSTANCE_NAME` directive. This means that each FME Server must be started with its own configuration file that specifies the desired instance name of that FME Server.

FME Server startups are specified as command line entries in the Process Monitor's configuration file (`<DDEInstallDir>/processMonitorConfig.txt`), and each FME Server's configuration file is specified as part of the command entry. For example, the following entries in processMonitorConfig.txt would start three FME Servers, each with their own configuration file. Each of these files would specify a unique instance name for its associated FME Server:

```
 CMDStartFMEServiceGreen="C:\\Program Files\\FME2004\\fme.exe"
REGISTER_SOCKET BRAD 7070 "C:\\Program Files\\SpatialDirect\\
fmeServerConfig_Green.txt"|log

 CMDStartFMEServiceYellow="C:\\Program Files\\FME2004\\
fme.exe" REGISTER_SOCKET BRAD 7070 "C:\\Program Files\\
SpatialDirect\\fmeServerConfig_Yellow.txt"|log

 CMDStartFMEServiceBlue="C:\\Program Files\\FME2004\\fme.exe"
REGISTER_SOCKET BRAD 7070 "C:\\Program Files\\SpatialDirect\\
fmeServerConfig_Blue.txt"|log
```

Note that each FME Server instance's configuration file can be named anything desired. The convention used in the example above uses a common `"fmeServerConfig_"` prefix with the instance name as a unique suffix to form the configuration file name. Each of these files would contain the `FME_INSTANCE_NAME` directive specifying the instance name to be `Green`, `Yellow` or `Blue` as appropriate.

Specifying Translation Acceptance Conditions

For each FME Server instance, one or more conditions can be specified that determine which translation requests are accepted by that server.

These acceptance conditions are specified in the QServer's configuration file, by default named `<DDEInstallDir>/qServerConfig.txt`. The conditions are specified in this file using a set of SLD `name=value` parameter entries.

Five types of acceptance conditions can be specified:

- Translation Complexity
- Selected Source Themes
- Selected Output Format
- FME Server Instance Name
- FME Server Host Name

Translation Complexity Conditions

Translation request complexity can be specified as a condition for acceptance by an FME Server instance. Please see *Default Request Filter* on page 221 for an explanation of translation complexity. Note that DDE request filtering does not need to be enabled for SLD to function.

Two forms of specifying the complexity condition are available: one for the specific FME Server instance and one for the host on which the instance is running. If both forms are specified, the instance complexities are used and the host complexities are ignored. The parameter syntax is:

```
INSTANCE_COMPLEXITY_RANGES=<fmeInstanceName>:<minComplexity or
*>-<maxComplexity or *>|<repeat for next instance...>
```

```
HOST_COMPLEXITY_RANGES=<fmeInstanceHostName>:<minComplexity or
*>-<maxComplexity or *>|<repeat for next instance...>
```

The asterisk denotes an "any" value. Here are some examples:

```
INSTANCE_COMPLEXITY_RANGES=Green:*-5.0|Yellow:5.1-9.9|
Blue:10.0-*
```

The above example specifies that FME Server instance `"Green"` will accept translation requests whose complexities lie between any minimum value and 5.0 inclusive. Requests whose complexities lie outside of this range will not be processed by FME Server instance `Green`. Similarly, instance `Yellow` accepts only requests whose complexities lie between 5.1 and 9.9 inclusive. Instance `Blue` accepts requests with complexities between 10.0 and greater.

```
   INSTANCE_COMPLEXITY_RANGES=Green:*-*|Yellow:12.0-*
```

The above example specifies that FME Server instance `Green` will accept translation requests of any complexity, while instance `Yellow` accepts complexities of 12.0 and greater.

```
HOST_COMPLEXITY_RANGES=bill:3.5-10.0|fred:10.1-*
```

The above example specifies complexity ranges in a manner similar to the instance examples above, but for FME Server hosts rather than for specific instances on those hosts. The example states that any FME Server instance running on host "`bill`" will accept (on a first-come, first-served basis) requests whose complexities lie between 3.5 and 10.0 inclusive. Similarly, any instance running on host "`fred`" will accept complexities of 10.1 and greater.

If an FME Server instance has no defined complexity range, it by definition accepts requests of any complexity. If a request itself has no defined complexity value, it can be accepted by any FME Server instance regardless of what complexity range may be defined for the instance. Note, however, that the request will still be rejected if it doesn't meet other acceptance conditions.

Selected Source Themes Conditions

The set of source themes selected for translation can be specified as a condition for acceptance by an FME Server instance. Two forms of specifying the source theme condition are available: one for only those source themes that are *accepted*, and one for those source themes that are always *rejected*. Either or both forms can be specified. The parameter syntax is:

```
INSTANCE_ACCEPTED_ONLY_THEMES=<fmeInstanceName>:<themeName
themeName...>|<repeat for next instance...>
```

```
INSTANCE_REJECTED_THEMES=<fmeInstanceName>:<themeName
themeName...>|<repeat for next instance...>
```

Here are some examples. Note that theme names must include the source data keyword suffix (such as "`_municipal`" in the example below):

```
INSTANCE_ACCEPTED_ONLY_THEMES=Green:streets_municipal
highways_municipal|Yellow:rivers_state lakes_state
```

```
INSTANCE_REJECTED_THEMES=Blue:railways_state
```

The above example specifies that FME Server instance `Green` will accept translation requests involving only the `streets_municipal` and/or `highways_municipal` source themes and no others. If a request includes any theme(s) other than either of these two specified themes, `Green` will not process the request at all. Similarly, instance `Yellow` only accepts requests for translating the `rivers_state` and/or `lakes_state` source themes and no others.

The second parameter specifies that FME Server instance `Blue` will always reject any request involving source theme `railways_state`, regardless of which other theme(s) may also be requested.

Note that entries in the accepted-only or always-rejected lists imply their converse presence in the other. That is, if `Green` accepts only the `roads` theme then by definition it always rejects all other themes. As a result, an instance name that occurs in one of the lists does not need to occur in the other list. SLD enforces this condition by removing any instance name entries in the always-reject list that are already present in the accept-only list, logging a warning to this effect in the QServer log file.

Selected Output Format

The selected translation output format can be specified as a condition for acceptance by an FME Server instance. Two forms of specifying the output format condition are available, one for only those output formats that are *accepted* and one for those output formats that are always *rejected*. Either or both forms can be specified. The parameter syntax is:

```
INSTANCE_ACCEPTED_ONLY_FORMATS=<fmeInstanceName>:<formatTagName
formatTagName...>|<repeat for next instance...>
```

```
INSTANCE_REJECTED_FORMATS=<fmeInstanceName>:<formatTagName
formatTagName...>|<repeat for next instance...>
```

Specifying Format Tag Names. The format tag names to use when specifying SLD output format conditions can be found in the `<format-tag>` second field of the entries in the `<DDEInstallDir>/safeViewerHTML/supportData/outputFormats.csv` file. Each tag name corresponds to an FME data format. Here are two sample entries taken from the file:

```
    2dwg12.fme,dwg,AutoCAD DWG (R12),dataFile
    2mapinfo.fme,tab,MapInfo TAB,dataDir
```

The first entry defines the AutoCAD R12 binary data format to DDE. The second field is “`dwg`” and this value would be used as the format tag name to specify the AutoCAD binary format in an SLD output format condition.

The second entry defines the MapInfo TAB data format to DDE. The second field is “`tab`” and this value would be used as the format tag name to specify the MapInfo TAB format in an SLD output format condition. Other formats are identified in the same way.

Here are some examples of SLD output format conditions:

```
INSTANCE_ACCEPTED_ONLY_FORMATS=Green:dwg dxf dgn|Yellow:e00 tab
INSTANCE_REJECTED_FORMATS=Blue:shp sdl
```

The above example specifies that FME Server instance `Green` will only accept translation requests for output formats `dwg, dxf` or `dgn` and no others. If a request specifies any output format other than one of these three specified formats, `Green` will not process the request. Similarly, instance `Yellow` only accepts requests for the `e00` or `tab` output formats and no others.

The second parameter specifies that FME Server instance `Blue` will always reject any request for output format `shp` or `tab`.

Note that entries in the accepted-only or always-rejected lists imply their converse presence in the other. That is, if `Green` accepts only the `dwg` output format then by definition it always rejects all other output formats. As a result an instance name that occurs in one of the lists does not need to occur in the other list. SLD enforces this condition by removing any instance name entries in the always- reject list that are already present in the accept-only list, logging a warning to this effect in the QServer log file.

FME Server Instance Name

A translation request can include an optional parameter indicating the name of the specific FME Server instance that should process it. When this condition is defined, only the named FME Server instance will process the request, even if other acceptance conditions are defined that would otherwise prevent the named FME Server instance from processing the request. In other words, the condition in which an FME Server instance is named by a request overrides any complexity, source theme or output format conditions that might be defined as well.

This acceptance condition can only be specified programmatically through the QServer API's `performTransaction` method. This method takes a Hashtable argument called `requestInfo`, and the desired FME Server instance name is added to the table as a value with the key `fmeInstance`. Please see the *`fmeInstance` parameter description* on page 163 for full information concerning this aspect of the QServer API.

The default DDE installation does not associate any FME Server instance name with any request. However, programmers wishing to interface with the QServer via its API (thereby bypassing the Translation Servlet) can use the API to require that specific FME Server instances should process specific translation requests.

FME Server Host Name

A translation request can include an optional parameter indicating the name of a specific FME Server host on which the request should be processed using an FME Server instance running on that host.

When this condition is defined, only the named FME Server host will process the request, using the first FME Server instance running on it that becomes available. This will occur even if other acceptance conditions are defined that would otherwise prevent the named FME Server host from processing the request. In other words, the condition in which an FME Server host is named by a request overrides any complexity, source theme or output format conditions that might be defined as well.

This acceptance condition can only be specified programmatically through the QServer API's `performTransaction` method. The method takes a Hashtable argument called `requestInfo`, and the desired FME Server instance name is added to the table as a value with the key `fmeHost`. Please see the *`fmeHost` parameter description* on page 163 for full information concerning this aspect of the QServer API.

The default DDE installation does not associate any FME Server host name with any request. However, programmers wishing to interface with the QServer via its API (thereby bypassing the Translation Servlet) can use the API to require that specific FME Server hosts should process specific translation requests.

Acceptance Condition Priority Order

SLD behavior can be fine-tuned to a high degree by specifying multiple acceptance conditions with appropriate values. In this regard, care should be taken to avoid defining complex conditions whose combination unintentionally blocks some translation requests from ever being processed.

When multiple conditions are specified, the QServer applies them in this descending priority order:

- If a request programmatically specifies an FME Server instance name, then only that instance will process the request. Any other conditions are ignored.
- If a request programmatically specifies an FME Server host name, then the request is processed only on that host by the first FME Server instance that becomes available. Any other conditions are ignored.
- Any complexity, source theme or output format conditions specified for a given FME Server instance must *all* be met by that instance for the instance to process the request.

  For complexity conditions, if both an instance complexity and a host complexity are specified for a given FME Server instance, the instance complexity is used and the host complexity is ignored. An FME Server instance's host complexity is used only if its instance complexity is not specified.

QServer Installation

The QServer application is contained in the `spatialDirect.jar` file in the DDE installation directory. The Java `CLASSPATH` environment variable must include the path name to this `.jar` file.

The following component is required:

- Sun Microsystems Java Runtime Environment (JRE) Version 1.4.1 or later.

This component is distributed as part of DDE.

QServer Configuration

Configuration parameters are used to set the various operating characteristics of the QServer. Values for these parameters are read from a configuration file when the QServer is started. The values assigned via the configuration file can be modified on a site-specific basis to suit the operating environment.

QServer Configuration File

The QServer configuration file is an ASCII text file called `qServerConfig.txt` that resides in the DDE installation directory. It consists of a collection of parameter assignments, each on a line. Each assignment consists of a parameter name followed by an equal-sign followed by the parameter's value. For example, the line:

```
REQUEST_PORT=7071
```

assigns the `REQUEST_PORT` parameter a value of `7071`.

Note: The QServer is case-sensitive to parameter names. Blank lines and lines beginning with the # character are treated as comments and are ignored.

The QServer must be restarted for changes to parameter values to come into effect.

The following parameters are included in the configuration file and must be named exactly as shown:

Parameter	Description	Default Value
`REQUEST_PORT`	The integer port number on which to listen for service transaction requests from requesters; for example, clients using the QServer API, such as the Translation Servlet. **Required**	7071
`SERVICE_REGISTRATION_PORT`	The integer port number on which to listen for services requesting to be registered as being available, such as FME Servers. **Required**	7070
`ADMINISTRATION_ REQUEST_ PORT`	The integer port number on which to listen for QServer administration transaction requests. **Required**	7072
`MAX_TRANSACTION_RESULT_ SUCCESSES`	Maximum number of successful result transactions to accept from the service before shutting down or restarting the service. **Required**	100
`MAX_TRANSACTION_RESULT_ FAILURES`	Maximum number of failed result transactions to accept from the service before shutting down or restarting the service. **Required**	10
`MAX_PENDING_CONNECT_ REQUESTS`	Maximum number of pending connect requests to allow on the `REQUEST_PORT`. Connect requests after this number will be rejected, rather than queued up for later processing. This is an internal TCP/ IP network socket limit only, and does not limit the number of pending transaction requests in the QServer's request queue. **Required**	50
`ARE_YOU_THERE_MSG`	The text string to send to a service to confirm that it is of the desired type before proceeding to use it. For FME Servers, this value must be `FME_AreYouThere`. **Required**	`FME_AreYouThere`

Parameter	Description	Default Value
`MAX_LOGFILE_LINES`	The maximum number of lines written to the current logfile, after which it is closed, followed by possible deletion of older ones and creation of a new one to continue on with. **Optional.** Values `<= 0` means size limiting is NOT in effect. The same log file is used and can grow indefinitely if logfile appending is in effect. Values `> 0` means size-limiting IS in effect using the number of lines specified. If absent, this parameter is assigned a default value of 3000 lines.	3000
`MAX_LOGFILE_AGE_SECONDS`	The maximum allowable age in seconds of previous versions of logfiles. Any logfiles older than this are deleted. Deletion of older logfiles only occurs when the current logfile exceeds `MAX_LOGFILE_LINES` in size and is closed. **Optional**. Values `< 0` means previous logfiles NEVER deleted. Values `= 0` means ALL previous logfiles deleted. Values `> 0` means previous logfiles older than specified value deleted. If absent, this parameter is assigned a default value of 604800 seconds (7 days).	604800
`SUCCESSFUL_RESULT_PREFIX`	The substring present at the start of the result string returned by a service to indicate a successful result. For FME Servers, this substring has a default value of `0:` (a zero followed by a colon). This default value can be overridden in the FME Server configuration file by using the `SUCCESS_RESPONSE` directive to include the desired prefix. *If the latter method is used, the prefix it specifies must also be the value assigned to this parameter.* **Required**	0:

Parameter	Description	Default Value
`SERVICES_USING_FAULT_ TOLERATOR`	Boolean flag indicating whether or not services are being run with the ProcessMonitor. A value of `true` requires that **all** services be run **with** the ProcessMonitor. `true` causes the QServer to shut down services that reach the `MAX_TRANSACTION_RESULT_SUCCESSES` or `MAX_TRANSACTION_RESULT_FAILURES` limits. Service restart is performed by the ProcessMonitor. A value of `false` requires that all services be run without the ProcessMonitor. `false` causes the QServer to shut down and restart services that reach the `MAX_TRANSACTION_RESULT_SUCCESSES` or `MAX_TRANSACTION_RESULT_FAILURES` limits. **Required**	true
`LOG_FILENAME`	The pathname of the message log file for the Qserver. Note that for Windows where the file-separator character is the backslash "\", the pathname value must specify the backslash as a **pair** of backslashes. **Required**	Site-specific: this will be set on installation.
`INCLUDE_TIMESTAMP_IN_LOG`	Boolean flag indicating whether or not to include a timestamp in the message log file. A value of `true` causes the timestamp to be included. Any other value causes it to be absent. **Required**	true
`INCLUDE_THREADNAME_IN_LOG`	Boolean flag indicating whether or not to include the name of the thread that logged the message in the message log file. A value of `true` causes the thread name to be included. Any other value causes it to be absent. **Required**	true

Parameter	Description	Default Value
`APPEND_TO_EXISTING_LOG`	Boolean flag indicating whether or not to append to the end of an existing message log file, if one exists, or to create a new one, overwriting, and thereby destroying, any same-named log file. A value of `true` causes messages to be appended. Any other value causes a new file to be created, destroying any previous one. **Required**	true
`ECHO_LOG_TO_CONSOLE`	Boolean flag indicating whether or not to display, or echo, all messages sent to the message log file to the default system output device, or console, as well. A value of `true` causes messages to be echoed to the console. Any other value prevents echoing. **Required**	false

Selective Load Distribution Parameters

In addition to the QServer configuration parameters listed above, the following optional parameters are also available to control the behavior of the Selective Load Distribution mechanism described in *QServer Selective Load Distribution (SLD)* on page 142.

Parameter	Description	Default Value
`ENABLE_SELECTIVE_LOAD_DISTRIBUTION`	Enables or disables Selective Load Distribution. **Optional.**	false
`INSTANCE_COMPLEXITY_RANGES`	Specifies translation complexity ranges for FME Server instances. **Optional**	none
`HOST_COMPLEXITY_RANGES`	Specifies translation complexity ranges for FME Server instance hosts. **Optional**	none
`INSTANCE_ACCEPTED_ONLY_THEMES`	Specifies source themes accepted by FME Server instances. **Optional**	none
`INSTANCE_REJECTED_THEMES`	Specifies source themes rejected by FME Server instances. **Optional**	none

Parameter	Description	Default Value
INSTANCE_ACCEPTED_ONLY_FORMATS	Specifies output formats accepted by FME Server instances. **Optional**	none
INSTANCE_REJECTED_FORMATS	Specifies output formats rejected by FME Server instances. **Optional**	none

QServer Start-up and Shutdown

The QServer is usually started automatically by the Process Monitor as part of the latter's sequenced start-up of all DDE components. This is the usual and recommended start-up procedure and is described in *Getting Started* on page 15. However, the QServer can also be started manually.

Manual Start-up

Manual start-up of the QServer is performed from the command line and has the following form:

```
java COM.safe.servicemanager.QServer <config>
```

where `<config>` is the full pathname of the QServer's configuration file. When the QServer is started manually from a command line, the `QServer>` prompt is displayed on the command line and waits for a command. Interactive QServer commands are described in *Interactive QServer Commands* on page 155.

The QServer must be started before any FME Servers are started, since upon start-up, each FME Server will attempt to register itself with the QServer.

Automatic Start-up

If the QServer is to be started automatically by the Process Monitor, the QServer startup command should be placed in the Process Monitor's configuration parameter file. This is the preferred method of starting the QServer.

Registering FME Servers with the QServer

Once the QServer has been started, one or more FME Servers can be started and registered with the QServer. FME Server start-up uses the `REGISTER_SOCKET` directive to register with the QServer, which is described in *FME Server Start-up and Shutdown* on page 128.

Note that the port on which the FME Server is registering must match the port specified by the `SERVICE_REGISTRATION_PORT` parameter in the QServer's configuration file.

Note If FME Servers are started with the Process Monitor, the FME start-up command should be placed in the Process Monitor's configuration parameter file. The QServer's configuration file has a parameter set to indicate whether or not its FME Servers are running under Process Monitor. If one FME Server is started with a Process Monitor, then all FME Servers must be started with Process Monitors. It is not possible to have some FME Servers running under a Process Monitor while others are running independently.

A manually started FME Server can be manually stopped by entering Ctrl+C on the same command line from which it was started.

Shutdown

It is recommended that the QServer be shut down as part of the overall DDE shutdown procedure. Please see Chapter 2 for details on DDE shutdown.

Less commonly, if the QServer was started manually on a command line, it can be shut down by issuing the interactive `stop` command at the QServer prompt. This interactive mode is useful for testing and debugging, but should not be used in production environments.

If the Qserver was started as a service, such as a Windows Service, then the shutdown mechanism provided by that service should be used to shut down the QServer.

QServer Logging

During operation, the QServer generates a log file to which it writes various messages indicating what has occurred and what it is currently processing. This is an ASCII text file that can be examined with any text editor.

The log file pathname is specified in the QServer's configuration file. If log file size-limiting is in effect, the actual pathname of the current log file is this base pathname plus an integer suffix representing the number of milliseconds since the Java epoch. If size-limiting is not in effect, the log file pathname is as-specified in the configuration file.

Interactive QServer Commands

The QServer can be used interactively by starting it manually from a command line. This interactive mode is useful for testing and debugging, but should not be used in production environments.

When started interactively, the `QServer>` prompt is displayed on the command line and waits for commands. The following table lists the available commands.

Command	Description
`pause requests`	This command pauses the request queue. Requests continue to be accepted and queued but are blocked on the queue.
`resume requests`	Resumes the request queue. Processing of requests on the queue then continues as usual.
`pause results`	Pauses the result queue. Results continue to be generated and queued but are held from being delivered to clients until the queue is resumed.
`resume results`	Resumes the result queue. Delivery of requests on the queue then continues as usual.
`show requests`	Displays information about each entry in the request queue.
`show results`	Displays information about each entry in the result queue.
`show queues`	Displays information about the state of the request and result queues.
`stop`	Stops the QServer.

Interfacing with the QServer

DDE is an integrated application. However, each of the components, including the QServer, may be interfaced with third-party applications. As the QServer represents the middle layer of the DDE hierarchy, a third-party application takes the role of the client that makes translation requests.

The QServer's QServerAPI forms the programmatic interface between the QServer and client applications such as the Translation Servlet. The API allows client applications to use the QServer to send client translation requests to available FME Servers, and to return translation result information back to the client.

The QServerAPI is straightforward to use and hides most low-level communication details from the client. The Translation Servlet uses this API to send requests and receive result status.

The QServerAPI is currently implemented in Java, C and C++.

This section is directed primarily towards developers wishing to interface their own client applications to the QServer. For purposes of illustration, the Java form of the API is described here, but the C and C++ forms are similar.

QServer API

The QServer API consists of four methods. The general approach involves creating a `QServerAPI` object and initializing it, sending the translation request to it, and then getting the result information back from it. The following sections describe the QServer API methods.

Construction

```
public QServerAPI()
```

This method is the parameterless constructor for the `QServerAPI` class.

Initialization

```
   public int init(String serviceManagerHost,
           int serviceManagerRequestPort)
           throws ServiceProviderException

   public int init(String serviceManagerHost,
           int serviceManagerRequestPort,
           String localHostIPAddress)
           throws ServiceProviderException
```

Both of these methods initialize the `QServerAPI` object. They return `kSPSuccess` if the `QServerAPI` object was successfully initialized, and throw a `ServiceProviderException` if initialization failed.

Transaction

```
   public int performTransaction(String requestKeyword,
                                 String request,
                                 Hashtable requestInfo)
                                 throws ServiceProviderException
```

This method sends the translation request to the QServer and blocks until the translation completes or an exception is thrown.

It returns a status value of `kSPSuccess` if the translation request was accepted by an FME Server and a translation result is available in the form of a `Transaction` object obtainable with the `getResultTransaction()` method.

The returned status value refers to the success or failure of the FME Server, rather than the status of the FME translation result itself.

Currently, the only status value returned by `performTransaction()` is `kSPSuccess`. All other conditions result in a `ServiceProviderException` being thrown when errors are detected, most being related to communication problems with the QServer.

Results

`public Transaction getResultTransaction()`

The `getResultTransaction()` method returns a `Transaction` object containing information describing the translation result. This information can be obtained by invoking various accessor methods on the returned `Transaction` object. The `Transaction` class is part of the `COM.safe.servicemanager` package.

In general terms, a `Transaction` acts as a wrapper for state information defining a transaction within the QServer. In the most general terms, a transaction consists of a client request to some service and the result returned by the service.

Requests and results both take the form of arbitrarily valued strings whose content is understood by both the client and the service. The QServer itself makes no attempt to interpret or validate the actual request or result content, other than to record and pass on the service's own indication of result success or failure.

The information of greatest interest in a `Transaction` is the translation result status. This is obtained as a string from a `Transaction` object using the `getResult()` method. The result string has the following default form:

```
<integer>:<message>
```

This default form can be overridden in the FME Server configuration file by specifying a different return value with the `SUCCESS_RESPONSE` and `FAILURE_RESPONSE` directives.

A successful FME translation has a result status string that by default starts with the following:

```
0:Translation Successful!
```

Unsuccessful FME translations have default result strings containing a non-zero <integer> value followed by the relevant error message text.

The `Transaction` object also contains an associated Boolean value indicating translation result status. This value is obtained using the `getResultSuccess()` method. The returned value indicates whether or not the translation was successful:

```
true = successful translation
false = unsuccessful translation
```

The following table lists all parameters currently available from a `Transaction` object and the methods used to obtain the value of each.

Parameter	Description	Accessor Method
`id_`	The unique integer ID of the translation request.	`long getID()`
`requesterResultPort_`	The local port on the client to which the result is to be sent. The client is listening on this port for the result.	`int getRequesterResultPort()`
`requesterHost_`	Host name string of the client to whom the result is to be sent.	`String getRequesterHost()`
`request_`	The original request string from the client that produced this result.	`String getRequest()`
`requestKeyword_`	The request keyword string from the client that accompanied the request.	`String getRequestKeyword()`
`result_`	The translation result string returned by the FME server. This string contains the status message of the FME translation, not the actual results.	`String getResult()`
`resultSuccess_`	A Boolean indicating the success or failure of the translation result.	`boolean getResultSuccess()`
`serviceMsg_`	A message string describing the status of the FME server operation itself, which is different from the translation `result_` returned by the FME server. The `serviceMsg_` indicates if an FME server succeeded in taking a `request_` and returning a `result_`, or if the FME server itself failed for some reason.	`String getServiceMsg()`

Parameter	Description	Accessor Method
`serviceSuccess_`	A Boolean indicating the success or failure of the FME server operation itself. This is different from the `resultSuccess_` Boolean which indicates the status of the `result_` returned by the FME server. The state indicated by the `serviceSuccess_` Boolean is described by the `serviceMsg_` string.	`boolean getServiceSuccess()`
`timeRequested_`	A Date object indicating when the translation request was received by the QServer and placed in the latter's request queue.	`Date getTimeRequested()`
`timeStarted_`	A Date object indicating when the translation request was taken from the QServer's request queue and sent to the FME server.	`Date getTimeStarted()`
`timeFinished_`	A Date object indicating when the translation result was received from the FME server and placed in the QServer's result queue.	`Date getTimeFinished()`
`otherParams_`	A Hashtable containing other, optional parameters describing the translation. The `priority` and `description` key-value pairs are stored here.	`Hashtable getOtherParams()`

QServerAPI Returns

In general, `QServerAPI` methods return an integer value of `kSPSuccess` for successful operation and throw a `ServiceProviderException` if an error occurs. For more details, please refer to Appendix D, *Status Return Codes*.

The `ServiceProviderException` class is defined in

```
COM.safe.serviceprovider.ServiceProviderException
```

Classes needing access to `QServerAPI` constants such as `kSPSuccess` can obtain them by including the following statement in their declarations:

```
implements COM.safe.serviceprovider.IServiceProviderConstants
```

Usage Details

1. Create a `QServerAPI`:

   ```
   QServerAPI qServerAPI = new QServerAPI();
   ```

2. Use either the 2-parameter or 3-parameter versions of the `init()` method to initialize the `QServerAPI`.

   The first version specifies the host name and request port number of the QServer system. The request port is the port on which the QServer is listening for service requests:

   ```
   int initStatus = qServerAPI.init("JIM", 7071);
   ```

   Specifying the Local Host IP Address in the init() Method

   The second version of the `init()` method specifies the same host name and port number but additionally allows the local host's IP address to be specified explicitly, rather than letting the QServerAPI determine it automatically as is usually the case:

   ```
   int initStatus = qServerAPI.init("JIM", 7071,
   "185.412.45.19");
   ```

   The local host IP address is the IP address of the system on which the QServerAPI is running, which can be different from the QServer system. The QServerAPI system referenced by this address is the system that receives the transaction result from the QServer.

   The address value can be specified as `null`, in which case the QServerAPI determines the address as described below.

   It is sometimes necessary to be able to specify the local host IP address explicitly. For example, when multiple IP addresses are defined for one machine and only one of them can be accessed by the QServer from behind a firewall, the accessible address would be the one specified in the `init()` call.

Specifying the Local Host IP Address on the Command Line

Alternatively, the local host IP address can be specified on the command line that starts the Java Virtual Machine to run the application that uses the QServerAPI. This is done by defining the address value as a Java system property called `localHostIPAddress`. Here is an example command line that does this:

```
java -DlocalHostIPAddress=”185.412.45.19" <appName> ...
```

This approach has the advantage of allowing the address value to be easily modified in the field to match changing conditions.

Determining the Local Host IP Address

The QServerAPI determines the local host IP address by making the following tests in the order shown:

- if the address is specified explicitly in the `init()` method, and its value is not `null`, then that address is used.
- otherwise if the address is specified as a system property on the command line, then that address is used.
- otherwise the address is determined automatically through the Java API.

3 Perform the translation, specifying the request keyword and request command strings and the request info hashtable:

```
int transactionStatus =
  qServerAPI.performTransaction(keyword,
command, requestInfo);
```

The `keyword` parameter identifies an optional configuration subsection defined in the FME Server's configuration file. The Translation Servlet application specifies either `SAFE_VIEWER_DISPLAY_RESULT` or `SAFE_VIEWER_NONDISPLAY_RESULT` as the values for this parameter.

The `command` parameter contains the actual FME command string that the QServer will send to an available FME server. Note that the API performs no validity checking for proper FME syntax on this command string. The string must make sense to the FME as-is.

The following example shows an FME command string generated by a client for a translation to GIF from two source data themes. FME command strings can differ to a great degree; this is an example of a representative command generated by the Translation Servlet:

```
2gif.fme  --THEMES "COASTL POLBNDL" --TimeStamp
941498194114 --MINX -140.0 --MAXX -109.0 --MINY 48.0 --MAXY
62.0 --DestCoordSys BPCNC --PIXELS 350.
```

The `requestInfo` parameter is a `Hashtable` containing additional information about the request. Any optional information about the request is supplied to the method in this hashtable.

This hashtable is examined for all optional information recognized by the method. All recognized information is extracted from the hashtable and incorporated into the request message string that is sent to the QServer. The latter supplies default values for certain optional parameters that are not explicitly specified.

Table Key	Table Value	Default Value (if not specified)
`timeRequested`	a Date object	now
`priority`	an Integer object	1
`description`	a String object	"no description"
`complexity`	a Double object	NaN
`outputFormatTag`	a String object	“undefined”
`fmeInstance`	a String object	no default defined
`fmeHost`	a String object	no default defined

- The `timeRequested` Date object specifies the time at which the QServer should submit the request for processing. The request is held until this time.
- The `priority` Integer object represents the request priority level. Higher values mean higher priority. Priority values must be `>= 0`. If a negative value is specified or the value is not a valid integer, the priority is set to 1. Note that priority values are instances of the Java Integer class, and not simple `int` primitives.
- The `description` string represents an arbitrary, client-defined description of the request.
- The `complexity` value represents the computed complexity of the request. This value is used by the QServer’s SLD mechanism when evaluating complexity-based acceptance conditions. In the standard DDE distribution, the Translation Servlet automatically computes and supplies this value.
- The `outputFormatTag` value identifies the output format that the request is asking for. This value is used by the QServer’s SLD mechanism when evaluating format-based acceptance conditions. In the standard DDE distribution, the Translation Servlet automatically supplies this value.
- The `fmeInstance` value identifies the name of the FME Server instance that should process the request. This value is used by the QServer’s SLD mechanism when evaluating instance name-based acceptance conditions. The standard DDE distribution does not specify any default value for this parameter.
- The `fmeHost` value identifies the name of the host on which an FME Server instance is running that should process the request. This value is used by the QServer’s SLD mechanism when evaluating host name-based acceptance conditions. The standard DDE distribution does not specify any default value for this parameter.

4 Get the result `Transaction` object back:

```
Transaction resultTransaction =
qServerAPI.getResultTransaction();
```

Code Example

This example code illustrates the use of the `QServerAPI` to send a translation request to the QServer and retrieve the result status:

```
// Create a QServerAPI and initialize it with the specified
// QServer host and port.

QServerAPI qServerAPI = new QServerAPI();
try
{
int initStatus = qServerAPI.init(qsHost_, qsRequestPort_);
}
catch(ServiceProviderException e)
{ // If QServerAPI initialization failed, output message and return.
System.err.println("Service Provider init failed with exception.");
System.err.println("Exception is: " + e.getMessage());
return;
}

// Have the QServerAPI perform a transaction. This method blocks
// until a result is returned or it throws an exception.
try
{ // Note that no additional transactionInfo is being sent.
Hashtable transactionInfo = new Hashtable();
int transactionStatus =
qServerAPI.performTransaction(transactionKeyword_,
transactionCmd_,
transactionInfo);
}
catch(ServiceProviderException e)
{ // If transaction performance failed, output failure message and return.
System.err.println(Transaction failed with exception.");
System.err.println("Exception is: " + e.getMessage());
return;
}

// If transaction performance was successful, get the result transaction.
Transaction resultTransaction = qServerAPI.getResultTransaction();


// Output some of the result transaction fields.
System.out.println("ID     = " +
Integer.toString(resultTransaction.getID()));
System.out.println("Result = " + resultTransaction.getResult());
```

DDE Translation Result Location

In DDE, the FME Server sets the location of the translation results it produces using the directives present in its configuration file.

In communicating the results back to the client, the FME Server includes the pathname of the result location that it used. This is usually accomplished by including the required pathname components in the value assigned to the `SUCCESS_RESPONSE` directive in the FME server's configuration file.

The pathname components ultimately appear at the client as part of the translation result string contained in the `Transaction` object. The client can obtain this string using the `getResult()` method and parse out the pathname components to determine how to access the actual translation results. This is the approach used by the Translation Servlet , which uses the components to construct a URL that a web browser can use to obtain the actual translation results.

CHAPTER

14

Translation Servlet

The Translation Servlet (referred to in this document as the *Servlet*) is a Java servlet program that accepts, validates and forwards translation requests from a web browser session or a client application via a URL interface to the QServer application. Using the result returned by the QServer, the Servlet creates an HTML page description which is returned to the calling application. If the client application can interpret HTML (such as a web browser) then the HTML page description can be displayed as a web page.

Theory of Operation

This section describes the theory of Servlet operation. Figure 14-1 provides an architectural overview of the Servlet and its components.

A client application sends a translation request in the form of a URL to the Servlet. For example, a browser-based client can send this URL as a web form submission to the Servlet, allowing the latter to obtain details of the request from fields in the form. Alternatively, a non-browser client can include translation request details within the URL itself, thereby supplying the Servlet with the necessary request information without requiring a web-based form.

The Servlet takes the submitted URL and converts its contents into a valid translation request that can be accepted by the FME Server. In a fully configured DDE system, the Servlet passes this request via the QServer to the FME Server.

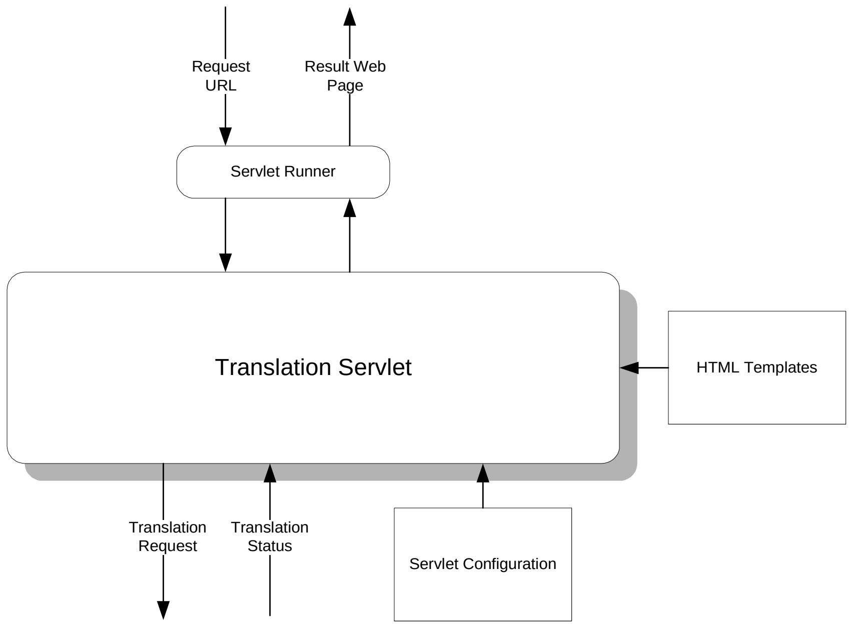


FIGURE 14-1 **Servlet Architecture**

After the FME Server has completed the translation, the result status is returned to the Servlet. An HTML template is retrieved, depending on the form of the original request. Embedded placeholders in the template are substituted with replacement values, such as the URL pointing to the translation result. The HTML page that results from this replacement operation is then returned to the client application. Depending on its purpose, the client can either display the web page or perform some other operation on the HTML stream.

In the default DDE configuration, the Servlet is started using either ServletRunner or Tomcat, as discussed in Chapter 16, *Servlet Engine Configuration*. However, site-specific customizations can change this, for instance, so that the Servlet is run by a web server plug-in, or directly by a servlet-capable web server.

Servlet Installation

The Servlet application is contained in the `spatialDirect.jar` file in the DDE installation directory. The Java `CLASSPATH` environment variable must include the pathname to this `jar` file. The following components are required:

- Sun Microsystems Java Runtime Environment (JRE) Version 1.4.1 or later.
- Sun Microsystems Java Servlet Development Kit (JSDK) 2.1 or later.

These components are distributed as part of DDE.

Additional Requirements

The Servlet needs to read a set of HTML template files that are part of the DDE distribution. By default, these files are copied into the web server's public HTML directory during DDE installation. When this is the case, the Servlet needs read access to the template files in the web server's public directory.

However, these template files can be relocated to a different directory if desired, as long as the Servlet has read access to them. The new location is made known to the Servlet by assigning the location's filepath value to the `htmlTemplatesDir` servlet property.

Servlet Configuration

The following sections give an overview of the considerations involved in Servlet configuration.

Servlet Properties

The default DDE configuration uses one of two servlet engines to run the Translation Servlet – Tomcat or ServletRunner – both of which are bundled with the DDE distribution. Each of these engines obtains servlet property values at start-up time from an engine-specific parameter file.

Full details of the parameter file and its modification are given in *Servlet Properties* on page 202.

Source Data Theme Definition

In the default DDE configuration, the Servlet obtains the definition of the source data themes from the following file:

```
<DDEInstallDir>\translationControl\system\config.csv
```

This file defines the characteristics of the source data themes. Each row in the file represents a source data theme and each field represents a parameter affecting an aspect of how the Servlet treats the presentation of the theme. Chapter 10, *Source Data Theme Definition* describes `config.csv` in detail.

The Servlet and the FME Server both make use of `config.csv` in the default DDE configuration. This file is always on the system upon which the FME Server executes. If necessary, the Servlet makes a local copy for itself during start-up, using the `layerListConfURL` servlet property to locate the file on the FME Server. See *Location of config.csv* on page 110 for more details regarding the location of `config.csv`.

Servlet HTML Pages

The `safeViewerHTML` directory tree contains HTML files used by the Servlet. These files can be modified to suit a local site's HTML requirements. Once these modifications have been made, the `safeViewerHTML` directory is copied into the web server's public HTML root directory. *HTML Template Placeholders* on page 197 discusses these template files in detail.

Servlet—FME Server Configuration

In the default DDE configuration, each FME Server is configured to work with the Servlet through the use of Servlet-specific subsections defined in the FME Servers' configuration files. FME Server configuration is described in *FME Server Configuration* on page 127.

There are two subsections defined in the default FME Server configuration file: `SAFE_VIEWER_DISPLAY_RESULT` for displayed output such as GIF and `SAFE_VIEWER_NONDISPLAY_RESULT` for non-displayed formats such as TAB, IGDS, Shape, and AutoCAD.

Servlet Start-up and Shutdown

The Servlet is usually started automatically by the Process Monitor as part of the latter's sequenced start-up of all DDE components. This is the usual and recommended start-up procedure and is described in Chapter 2, *Getting Started*.

The Servlet runs as a multi-threaded application. Each browser or URL request to the Servlet causes a new thread of execution to be started to handle that request, enabling multiple requests to be concurrently processed.

For full details on Translation Servlet start-up and shutdown, refer to Chapter 16, *Servlet Engine Configuration*.

Servlet Logging

During operation, the Servlet generates a log file to which it writes various messages, indicating what has occurred and what it is currently processing. This is an ASCII text file that can be examined with any text editor.

The log file pathname is specified in the servlet properties file. If log file size-limiting is in effect, the actual pathname of the current log file is this base pathname plus an integer suffix representing the number of milliseconds since the Java epoch. If size-limiting is not in effect, the log file pathname is as-specified in the properties file.

Log-in Sessions

The Servlet supports a basic user log-in mechanism. This is used to retain certain user-specific settings or properties. These properties are stored between log-in sessions in user property files. The Servlet restores these properties for the given user the next time they log in.

Note that log-in sessions are used only for this purpose — to identify a user so that their property settings can be restored. Log-in sessions are not used as a security mechanism and do not control or restrict access to DDE.

User Properties

User properties files are stored in a directory specified by the Servlet property `preferencesDir`. (User properties are distinct from and independent of Servlet properties). They have filenames with a prefix consisting of the user's name, and a file extension of `.prp`. For example, a user called `cat` would have a user property file called `cat.prp`.

The log-in property files are plain ASCII text files and can be modified with a text editor.

User property files contain various user-specific properties, such as user name, as well as settings used for the most recent translation request, such as theme list selections, search area boundaries and query method.

Two properties are always present in the user property file, the name and password properties. Although the current Servlet implementation does not make use of the password, the system still defines and maintains them for future use. Later versions of DDE may support the concept of user names with associated passwords.

User Property File Contents

A user property file contains two comment lines prefixed by the comment hash character `#` that identifies the user and the file's creation date, followed by a `<property-name>=<property-value>` pair on each line. The order that properties appear in the file is not important.

Here is a partial listing of an example user property file called `cat.prp`:

```
#Properties for cat
#Wed Nov 01 16:09:01 PST 2000
zipResult=FME_b003121144390851.zip
scale=%scale%
pass=cat
notificationEmailAddress=cat@somewhere.com
upperRightY=56.74875000000001
upperRightX=-109.96875
resultsDeletionTime=Wed-01-Nov-2000\ 05\:07\:26\ PM
mappingFilename=2gif.fme
panFactorPercentIndex=3
resultsTempDir=http\://NOODLE\:80/results
pixels=350
enableAutoThemeSelection=false
coordSystem=BCPOLY-83
gifResult=FME_b003121144390758.gif
minY=54.41575
minX=-113.84375
retainThemeListSelections=true
maxY=56.74875000000001
maxX=-109.96875
zoomFactorIndex=0
lowerLeftY=54.41575
lowerLeftX=-113.84375
servletURL=http\://NOODL\:8194/servlet/translationServlet
queryMethod=dbUnits
coordsys=BCPOLY-83
selectedThemes=COASTL\ POLBNDL\ BUILTUPA\ MISPOPP
name=cat
fmeLog=FME_b003121144390758.log
```

Restored Settings

The following settings are restored for each user log-in session:

- search area query method
- search area bounds
- coordinate system selection
- theme list selections

The first three settings are always restored between sessions, while the theme list selection is restored depending on the value of the user property `retainThemeListSelections`. If this property is present in the user's property file and has an assigned value of `true`, then the theme list selections will be retained and restored between sessions.

If the property is absent, or has any value other than `true` assigned to it, the theme list selections will not be retained between sessions. This property is added to each property file by the Servlet and set to an initial default value of `true`. It can be subsequently changed to `false` with a text editor if desired.

Log-in Processing Sequence

When a user logs in to start a session, the Servlet checks for the existence of a user property file with a file name equal to the log-in ID of the user. For example, if a user logs in with the name of `generaluser`, the Servlet searches for a property file called `generaluser.prp`. If the file is found, the property values it contains will be read in and used during the session. If the file is not found, it will automatically be created with the appropriate name along with a set of initial default property values.

Default Web Pages

The Servlet is distributed with a number of default web pages that allow a user to interact with DDE via a browser. The following section describes the default query form web page.

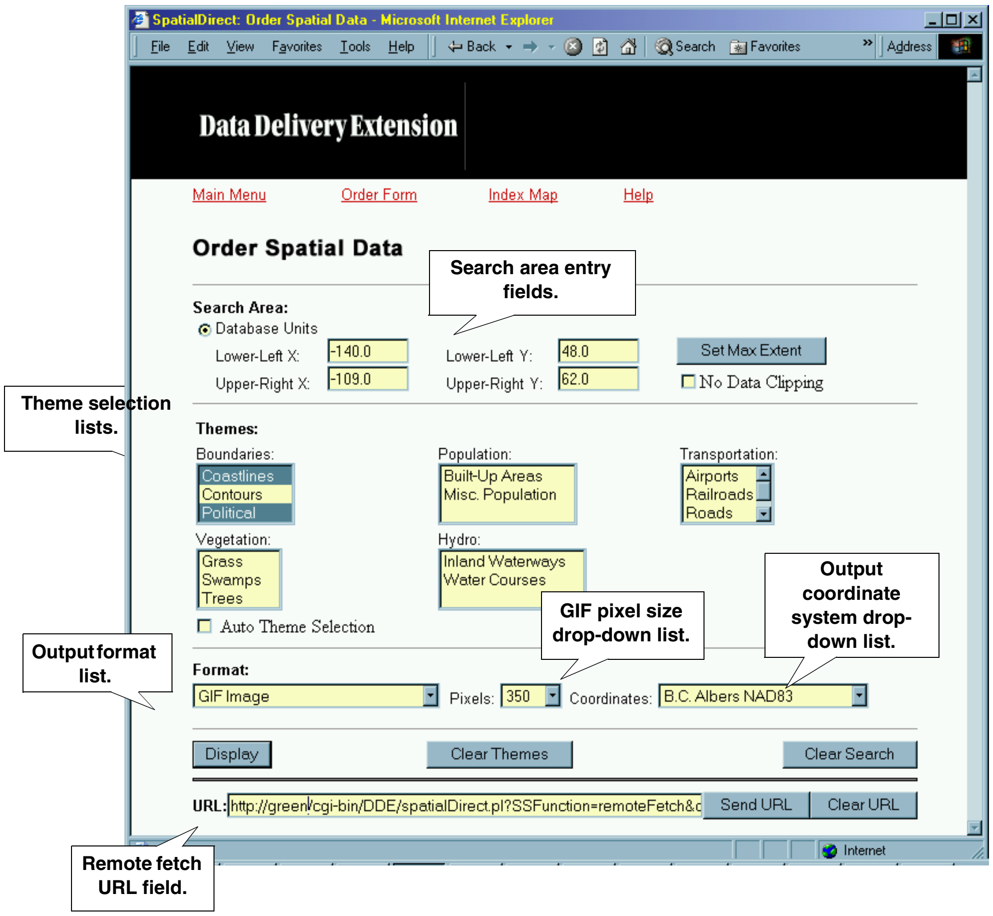
Query Form Web Page

DDE comes bundled with a browser web page which allows the user to query the source dataset and obtain the result in a specified format.

The web page consists of a form into which values are entered by the user, each one modifying the form of the query. When the display/order button is clicked, the web page form's content is passed to the Servlet which generates the required translation command and sends it to the FME Server. The latter translates the source data according to the command and returns the result status to the Servlet, which sends the appropriate results web page to the browser.

Entry Fields

The Servlet's default query form web page is shown in Figure 14-2. The entry fields are annotated on the form and are discussed in detail below. The values shown entered into the form are from the sample dataset.


Search area entry fields.
Theme selection lists.
Output format list.
GIF pixel size drop-down list.
Output coordinate system drop-down list.
Remote fetch URL field.

SpatialDirect: Order Spatial Data - Microsoft Internet Explorer

File Edit View Favorites Tools Help | Back | Search | Favorites | Address

Data Delivery Extension

Main Menu   Order Form   Index Map   Help

Order Spatial Data

Search Area:
(●) Database Units

Lower-Left X:	-140.0	Lower-Left Y:	48.0	Set Max Extent
Upper-Right X:	-109.0	Upper-Right Y:	62.0	☐ No Data Clipping

Themes:

Boundaries: Coastlines, Contours, Political
Population: Built-Up Areas, Misc. Population
Transportation: Airports, Railroads, Roads
Vegetation: Grass, Swamps, Trees
Hydro: Inland Waterways, Water Courses

☐ Auto Theme Selection

Format:
GIF Image   Pixels: 350   Coordinates: B.C. Albers NAD83

Display   Clear Themes   Clear Search

URL: http://green/cgi-bin/DDE/spatialDirect.pl?SSFunction=remoteFetch&c   Send URL   Clear URL

Internet

FIGURE 14-2 **Default Query Form Page**

The form contains the following data entry fields:

- **Search Area:** The Search Area fields contain the bounds of a rectangle used for searching the database. The query will return all data contained in the selected themes within the search rectangle (however, see *No Data Clipping* below). The rectangle is entered as four values representing the lower-left (x,y) and upper-right (x,y) coordinates.

  These coordinate values must be expressed in the coordinate system defined by the `QueryCoordSys` macro as described in *Specify Query Coordinate System* on page 117. The `Set Max Extents` button will set these values to cover the entire area of the source dataset. Note that these values come from the Servlet properties files. If the user attempts to enter values outside of the valid ranges of the extent, then an appropriate error message is displayed.

If the **No Data Clipping** checkbox (not shown in the figure) is checked, data clipping is not performed by DDE after it receives the features from the source dataset. (However, any initial clipping that might be performed by the underlying data source system itself is retained.)

In the default DDE environment, source dataset databases such as SDE return to DDE all features lying within and crossing over the specified search rectangle bounds *in their entirety.* By default DDE then clips the features to the specified search rectangle, truncating any crossover features at the rectangle bounds. Checking the **No Data Clipping** checkbox prevents DDE from performing this clipping operation, leaving any features that cross over the bounds whole, intact and extending beyond the bounds as necessary.

Note One caveat to not truncating crossover features: if very long linear features or very large polygonal features exist that just barely enter the search rectangle, the *entire* very long/large feature along with the other features within the rectangle will be returned. Depending on their relative sizes, scaling to see the very large, untruncated features may force the majority of features to become very small.

- **Themes:** The themes represent the logical partitioning of features in the underlying source dataset. Usually all features in a given theme share the same geometric and attribute schema. The theme lists that are displayed are generated from the `config.csv` file, described in Chapter 10, *Source Data Theme Definition*.
- **Format:** The desired output data format is selected from this pull-down list. The items appearing in this list are determined by the content of a configuration file maintained for this purpose.
- **Pixels:** The Pixels field represents the height and width of the image in pixels when the output result is to be a displayed GIF image.
- **Coordinates:** The desired destination coordinate system of the result is selected from this pull-down list. The items in this list are determined by the content of a configuration file maintained for this purpose.
- **URL:** This input field is a tool to assist developers in the construction of URL strings. As the various controls on the query window are altered, the URL string is updated in real time. This represents a URL-encoded form of the translation request as specified in the order form fields above.

  The URL can be sent to the Servlet by clicking the **Send URL** button. This initiates the *remote fetch* form of the translation request, and is equivalent to clicking on the **Display/Order** button. The URL string can be cleared by clicking the **Clear URL** button.

  A URL can also be manually entered and sent. If the URL conforms to the DDE syntax for remote fetch URLs, it will return the expected results. Otherwise, a web page indicating the URL error will be displayed.

In production environments, this URL input field and its associated buttons can be removed from the web page by modifying the relevant HTML code.

Servlet Processing Functions

Each URL request sent to the Servlet specifies the processing function the Servlet is to perform. This is done by assigning the appropriate value to the `SSFunction` parameter within the URL (or within the form as a hidden variable).

For the most part these values are set and used internally by the various default Servlet HTML web pages, and need not concern the end user. However, use of the `remoteFetch` function value is required in order to programmatically interface with the Servlet. Its use is described on *Remote Fetch URL Interface* on page 177.

The table below lists the possible values that `SSFunction` may take.

TABLE 14-1 SS Function Values

SSFunction Value	Servlet Function
`prepareFetch`	Displays order form page.
`remotePrepareFetch`	Displays order form page for integrations via remote fetch URL.
`remoteFetch`[1]	Gathers translation parameters, does user log-in, then performs `autoFetch`.
`autoFetch`	Gathers translation parameters, then performs `fetch`.
`fetch`	Creates and sends translation request for specified format.
`reFetch`	Creates and sends translation request for different specified format.
`zoompan`	Creates and sends translation request based on zoom/pan settings.

1. Used in remote fetch URLs.

Remote Fetch URL Interface

Non-browser applications external to DDE can interface with the Servlet and send it translation requests by using the *remote fetch URL* mechanism.

This allows a completely non-interactive translation request to be performed by sending a specific form of URL to the Servlet in what is known as a remote fetch.

The remote fetch URL can include all, some or none of a number of parameters required to perform the translation. Any parameters not explicitly supplied by the URL are supplied with default values by the Servlet.

There are two methods of sending remote fetch URLs: directly to the Translation Servlet engine or indirectly through the webserver via a DDE CGI Perl script which forwards the request to the engine, awaits the response and sends it back.

The indirect method allows requests to reach DDE when the latter is operating behind a firewall and only the default webserver port is available to receive requests. The CGI Perl script is called `spatialDirect.pl` and is copied during installation into a subdirectory (user-defined during installation) within the webserver's main CGI scripts directory.

To be processed as a remote fetch, a request URL must assign a parameter called `SSFunction` a value of `remoteFetch` and must have at least one of the two following minimum forms:

Directly to the Translation Servlet Engine

```
http://<translationServletHostName>:<portNum>/servlet/
translationServlet?SSFunction=remoteFetch
```

Here is an example of a minimum remote fetch URL sent directly to the Translation Servlet engine:

```
http://JIM:8194/servlet/
translationServlet?SSFunction=remoteFetch
```

When this URL is sent to the Servlet, the latter will return a display of the results of its automatically generated default query.

Indirectly through the WebServer

```
http://<webserverHostName>/<cgiBinDir>/<sdScriptDir>/
spatialDirect.pl?SSFunction=remoteFetch
```

Here is an example of a minimum remote fetch URL sent indirectly to the Translation Servlet engine via the forwarding Perl script run by the webserver:

```
http://JIM/cgi-bin/SpatialDirect/
spatialDirect.pl?SSFunction=remoteFetch
```

When this URL is forwarded to the Servlet, the latter will return a display of the results of its automatically generated default query.

URL Syntax Notes

In order to correctly specify values in a remote fetch URL, certain characters such as double-quote, space, etc, need to be specified as hex numbers or as alternate characters in order for the web server to correctly parse them. Complete lists of these encoded characters can be found in general HTML reference books. Here are some examples of these character encodings:

Character	Meaning	Encoding
“	double-quote	%22
<	less-than	%3C
=	equals	%3D
	space	+

For example, to include the following macro definition in the remote fetch URL:

```
--SDE30WhereClause "width < 20 AND numLanes = 4"
```

the definition would need to appear in the URL as:

```
--SDE30WhereClause+%22width+%3C+20+AND+numLanes+%3D+4%22
```

Note the use of the plus signs (+) to represent space characters within the URL. Also note the use of the `%22` pair (double-quotation-mark encoding) bracketing the entire macro expression value. Because the value contains spaces, enclosing it in double quotation marks is necessary so that FME uses the entire expression as the value for the macro.

Remote Fetch URL Parameters

The table below lists the additional parameters that can be explicitly specified in the remote fetch URL if desired:

TABLE 14-2 Remote Fetch URL Parameters

Parameter	Default Value	Description
`coordsys`	`LL-27` (lat/long NAD27)	output coordinate system
`description`	“`no description`”	user-defined text string describing request. *(see parameter notes below)*
`enableAutoThemeSelection`	set in Servlet’s `servlets.properties` file	`true` or `false` to set auto theme selection
`fmeParams`	*none*	user-defined custom FME command string. *(see parameter notes below)*

TABLE 14-2 Remote Fetch URL Parameters

Parameter	Default Value	Description
`fmeServerSubsection`	*none*	specifies the name of a user-defined custom subsection in the FME Server's config file that the Server will use for the translation
`format`	`2gif.fme`	name of FME mapping file
`lowerLeftX`	the `minXSearchLimit` value	minimum x-coordinate of search area
`lowerLeftY`	the `minYSearchLimit` value	minimum y-coordinate of search area
`name`	`anonymous`	log in user name *(see parameter notes below)*
`notificationEmailAddress`	*none* (blank address)	default e-mail address to send asynchronous result notifications to *(see parameter notes below)*
`pass` *(currently unused)*	*none*	log in user password *(see parameter notes below)*
`pixels`	`350`	pixel size of GIF output
`priority`	1	request priority used by QServer *(see parameter notes below)*
`queryMethod`	`dbUnits`	query form search area method
`returnTemplateFilePrefix`	*none*	prefix of the HTML template files that will be used after the translation request has completed *(see parameter notes below)*
`scale` *(currently unused)*	`WC50_`	scale prefix
`timeRequested`	*immediately*	time the request is to be submitted by QServer for processing *(see parameter notes below)*
`upperRightX`	the `maxXSearchLimit` value	maximum x-coordinate of search area
`upperRightY`	the `maxYSearchLimit` value	maximum y-coordinate of search area

TABLE 14-2 Remote Fetch URL Parameters

Parameter	Default Value	Description
`userSelectedThemes`	set in `config.csv` file	user-selected themes
`willWaitForDelayedResults`	`false` (client will not wait)	flag indicating client will wait synchronously for result delivery *(see parameter notes below)*

Parameter Notes

description

The `description` specifies an arbitrary description string used to identify or otherwise describe the request.

name

Because remote fetches have log-in performed for them by the Servlet, remote fetch URLs can optionally include the log-in user name. This is specified with the following parameter syntax:

```
name=<username>
```

If this parameter is present in the remote fetch URL, the Servlet will automatically log in the request using the specified user. If absent, the request will be logged in using the default user name `anonymous`.

notificationEmailAddress

Specifies the e-mail address to which the QServer should send result notifications.

Example: `notificationEmailAddress=user@company.com`

pass

The `pass` parameter is used to specify the user log-in password. Although this is defined and accepted by the Servlet, it is not currently used.

priority

Priority values must be integers greater than or equal to 0, with higher values meaning higher priority. There is no logical restriction on the maximum value. If a request's priority value is less than 0 or is not specified, the QServer sets it to 1.

Requests with higher priority are inserted into the request queue ahead of requests with lower priority. If requests with the same priority as a new request are already present, the new request is inserted after the last of those requests.

Example: `priority=4`

returnTemplateFilePrefix

If a remote fetch URL specifies the `returnTemplateFilePrefix` parameter, it is indicating that a user-supplied custom template file should be used to return translation results, rather than the default-supplied template files.

This URL parameter specifies only the prefix portion of the user's custom return template file. There can be two such files, one for successful translations and one for failed ones. Both files must be located in the `<webServerPublicRootDir>\safeViewerHTML\htmlTemplates` directory, and must be named using the following syntax:

```
<prefix>Success.html
<prefix>Failure.html
```

For example, if the remote fetch URL specified a return template file prefix of `dog`, there would be user-supplied files called `dogSuccess.html` and `dogFailure.html` in the `htmlTemplates` directory.

If either of the files exists, it is used in place of the default-supplied template files wherever appropriate. If either of the files is absent, or if the `returnTemplateFilePrefix` parameter is absent in the remote fetch URL, the default-supplied template file is used instead.

timeRequested

Specifies the date/time stamp at which the QServer should submit the request for processing. The date/time value must use the following Java time format:

```
E-dd-MMM-yyyy+hh:mm:ss+a
```

Example: `timeRequested=Mon-10-Apr-2000+10:12:00+AM`

Note that plus signs are used where spaces would normally go. This must be done to conform to URL syntax requirements.

Also note that this format is 12-hour–based, with `AM` or `PM` needing to be specified.

userSelectedThemes

The default `userSelectedThemes` are the initial display themes as set in the `config.csv` file.

willWaitForDelayedResults

Specifies whether or not the client (the sender of the remote fetch URL) will wait for complex and/or delayed results or expects notification e-mail to be sent instead.

The value assigned must be either `true` or `false`. If this parameter is absent or has any value other than `true`, `false` is assumed and e-mail notification will be sent for complex and/or delayed requests.

Note that when the value of this parameter (or its absence) causes notification e-mail to be sent, the `notificationEmailAddress` parameter should also be specified.

Example: `willWaitForDelayedResults=true`

lowerLeftX, upperRightX, lowerLeftY, upperRightY

The default minimum and maximum `x` and `y` search area values that are used are those specified as the search limits in the Servlet's properties file.

fmeParams

An optional parameter called `fmeParams` can also be included in the URL if desired.

This parameter's value can be set to a custom string having user-defined content which the Servlet appends to the FME translation command string. Such a string would typically contain additional FME directives and/or translation-specific macro definitions. Macro definitions must follow the usual FME syntax, which requires that the macro name be prefixed by two hyphens (--).

Note that the Servlet performs no validity checks or interpretation on this string – it must make sense to the FME as-is and is simply appended to the FME command string by the Servlet.

The following table lists the predefined FME macros that can be specified in the `fmeParams` parameter. Note that *any* macros or valid directives can be specified using `fmeParams`, not just the ones listed in the table. The only consideration is that the macros or directives, to be effective, must be correctly referenced within the mapping file(s) used by FME for the translation. The table lists macros that are referenced and used by the standard DDE mapping files and are therefore available for use.

Macro Name	Default Value	Description
`CLIP`	`yes`	Enables/disables clipping by the FME Server. *See usage notes below.*
`ClippingCoords`	min/max x/y values of default query clipping rectangle	x/y coordinate list of points defining query clipping polygon. *See usage notes below.*
`configFile`	`config.csv`	Name of main theme configuration file.
`dotSize`	`3`	Size of GIF/PNG point features expressed in number of pixels. Affects GIF/PNG displayed output only.
`GifLabelColor`	`Black`	All GIF/PNG label text is set to this color. Affects GIF/PNG displayed output only.
`GifMaxNumLabels`	`150`	Max number of labels allowed for a theme output in GIF/PNG. If theme has more than this number of labels, then *no* labels are displayed for this theme. Affects GIF/PNG displayed output only.
`IMAGEMAP_THEME`	`undefined`	Name of single theme that will have a clickable image map in GIF/PNG output. Affects GIF/PNG displayed output only. *See usage notes below.*
`LabelSizeMultiplier`	`0`	Floating point number `>= 0.` All label text sizes will be multiplied by this value if macro is present and defines a non-zero value. If macro is absent or its value is `0`, text sizes remain at the `TextHeight` value specified in `config.csv`.
`QueryCoordSys`	`LL84`	FME name of coordinate system in which coordinates of query clipping polygon are expressed. Can be different from source data coordinate system(s).
`SDE30SearchFeature` *(for SDE source format only)*	same coordinates as default rectangular search envelope	SDE search feature coordinates as a set of (x,y) coordinate pairs defining the geometry of the search feature. *See the FME Readers and Writers manual for details.*
`SDE30SearchMethod` *(for SDE source format only)*	`SDE_ENVELOPE`	SDE search method keyword. *See the FME Readers and Writers manual for list of available SDE search method keywords.*
`SDE30WhereClause` *(for SDE source format only)*	"1 = 1" (all features included)	SDE SQL "Where" clause. *See the FME Readers and Writers manual for details.*

`fmeParams` Usage Notes

CLIP

The value of `CLIP` must be `yes` or `no`. The default is `yes`. Clipping is performed using by default the rectangle defined by the values of the `MINX`, `MAXX`, `MINY` and `MAXY` macros.

Specifying `yes` for `CLIP` outputs features fully inside the query clipping polygon as well as the inside portions of features that are clipped. Specifying `no` prevents clipping from being performed by the FME Server.

ClippingCoords

The `ClippingCoords` value must be a space-delimited sequence of x-y coordinate pairs defining a closed polygon. The last point must be a repeat of the first point in order to close the shape, and the coordinate sequence must travel around the polygon's perimeter (clockwise or counterclockwise) without crossing the interior. The polygon's coordinates must be in the coordinate system defined by DDE as being the coordinate system for query polygons.

Here is an example of a `ClippingCoords` macro that specifies a Lat/Long triangular clipping region in the default sample dataset that ships with DDE:

```
fmeParams=--ClippingCoords+%22-124.5+48.0+-
120.625+50.333+-120.625+48.0+-124.5+48.0%22
```

Note the required syntax using plus signs (+) for spaces and the `%22` special-character encoding representing double quotes bracketing the entire coordinate list. Because the list contains spaces, double-quoting it is necessary to cause FME to use the entire list as the value for the `ClippingCoords` macro.

IMAGEMAP_THEME

If the `IMAGEMAP_THEME` value is undefined, all themes whose `GifAlt` attribute (defined in `config.csv`) is not "-" get a clickable image map. This is the default. If the `IMAGEMAP_THEME` value is set to a specific theme name, only that single theme (whose `GifAlt` value cannot be "-") gets a clickable image map. Please see *GIF Image Display Characteristics* on page 264 for more information.

DDE Example Remote Fetch URL

Here is an example of a remote fetch URL that explicitly specifies values for some of the available parameters, including `fmeParams` to define some custom FME macros:

```
http://JIM:8194/servlet/
translationServlet?SSFunction=remoteFetch&
name=bill&
format=sde2gif.fme&
pixels=700&
queryMethod=dbUnits&
coordsys=LL-27&
userSelectedThemes=CITY_AREA+STREET&
lowerLeftX=3436000&
lowerLeftY=3450000&
upperRightX=3500000&
upperRightY=3514000&
enableAutoThemeSelection=true&
fmeParams=--NUM_LANES+4+--TREE_TYPE+deciduous
```

Some of the parameter values contain + characters. Each of these represents a space wherever it occurs in the value and must be used instead of actual space characters. The + characters are replaced by spaces once the value arrives at the Servlet.

Customizing the Query Form Page

The Servlet's Query Page Handler Interface is a Java Interface that declares methods used to process a portion of the Servlet's query form page.

The Interface is called `IViewerQueryPageHandler` and is part of the `COM.safe.viewerservlet` package. This Interface was defined to support the development of customized query form pages with the Servlet.

This section is directed primarily towards developers wishing to write custom Java query handlers and HTML query pages for use with the Servlet.

Background

The query form page sent by the Servlet for display in the browser contains various fields for entering parameters describing the desired query. Some of these parameters are specific to a given query environment, while others are common to all environments.

Application common query parameters are processed directly by the Servlet.

However, the Servlet processes application specific query parameters using a custom handler written specifically for this purpose. The handler is implemented as a Java class that implements the `IViewerQueryPageHandler` interface.

Custom Query Form Pages

The default query form page supplied by the Servlet supports general request operations and can be used in many cases as it stands. However, specialized query form pages can be developed and used in place of the default.

Developing a custom query page involves creating the HTML file defining the form page and writing a custom Java query page handler class to operate with the companion HTML form.

At run-time the Servlet dynamically loads the required handler class and uses it to communicate with the associated query form page. Servlet properties are used to set the required handler and HTML page.

The HTML file for the default form is used as a starting template, since it contains code that is required for all query form pages.

The query page handler class must implement a number of methods defined in a Java interface called `IViewerQueryPageHandler`. These methods primarily require the handler to obtain current information from its companion form and send it back to the Servlet.

Query Handlers and Query Pages

When the Servlet is first started, it dynamically loads the handler's class definition and creates a single instance of the handler. The Servlet then uses the custom handler to process application-specific query parameters by invoking on it the various `IViewerQueryPageHandler` methods implemented by the handler class.

There is a tight logical coupling between the query page handler code and the HTML code in its associated query web page file. The handler and the web page file act as a pair to process the application-specific query parameters.

The Servlet must be told which query handler and associated web page file to use. This is accomplished by setting the values of two Servlet properties: `queryPageHandlerClassName` and `queryPageHandlerHTMLFile`.

The handler class definition must be accessible via the Java CLASSPATH environment variable, and the web page file must be located in `<webserver-html-root-dir>\safeViewerHTML\htmlTemplates`.

The Servlet is distributed with a default query handler and web page file. The default handler's class is

```
COM.safe.viewerservlet.DefaultQueryPageHandler
```

The default web page file is `fetchDefault.html`.

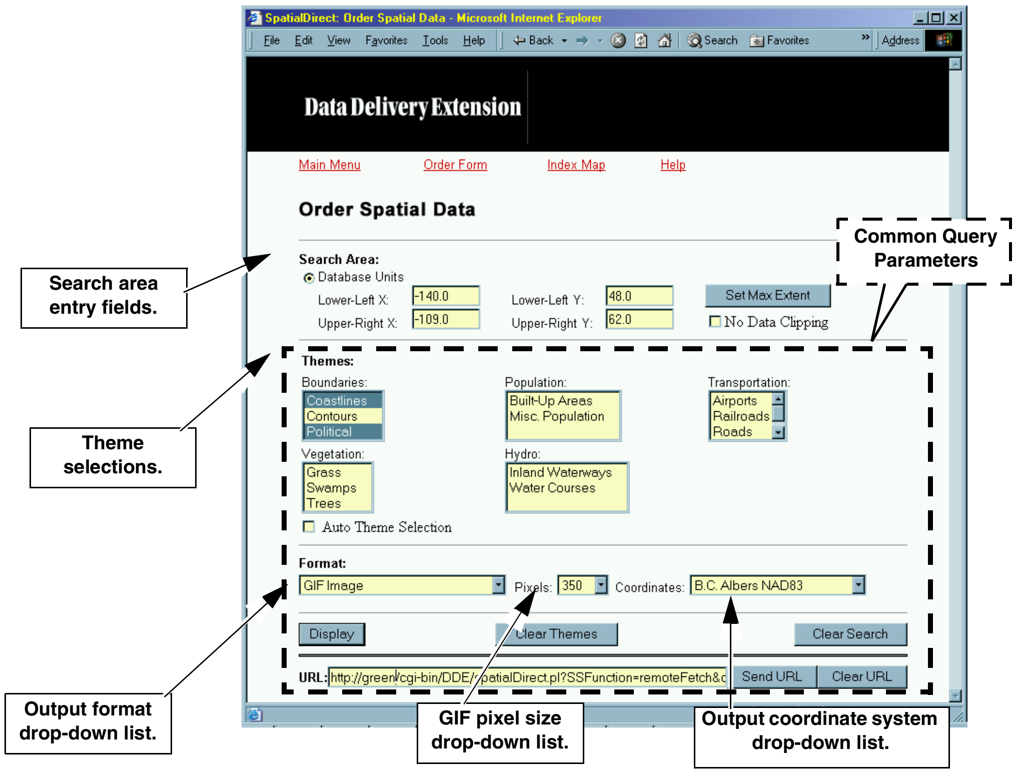
The source code for this default handler is supplied as part of this distribution. It can be used as a reference sample when developing a custom query page handler. This source code file is supplied in:

```
<webserver-html-root-dir>\safeViewerHTML\htmlTemplates\
DefaultQueryPageHandler.java
```

The default web page file is supplied in:

```
<webserver-html-root-dir>\safeViewerHTML\htmlTemplates\
fetchDefault.html
```

Figure 14-3 illustrates the query form page that results when using the default handler and web page.


Search area entry fields.
Theme selections.
Output format drop-down list.
GIF pixel size drop-down list.
Output coordinate system drop-down list.
Common Query Parameters
SpatialDirect: Order Spatial Data - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Address
Data Delivery Extension
Main Menu Order Form Index Map Help

Order Spatial Data

Search Area:
● Database Units

Lower-Left X: -140.0 Lower-Left Y: 48.0 Set Max Extent
Upper-Right X: -109.0 Upper-Right Y: 62.0 ☐ No Data Clipping

Themes:

Boundaries:	Population:	Transportation:
Coastlines	Built-Up Areas	Airports
Contours	Misc. Population	Railroads
Political		Roads

Vegetation:	Hydro:
Grass	Inland Waterways
Swamps	Water Courses
Trees	

☐ Auto Theme Selection

Format:
GIF Image Pixels: 350 Coordinates: B.C. Albers NAD83

Display Clear Themes Clear Search

URL: http://green/cgi-bin/DDE/spatialDirect.pl?SSFunction=remoteFetch&c Send URL Clear URL

FIGURE 14-3 **Servlet Default Query Form**

Customizing the Query Page

A number of Servlet query parameters are common to all applications. They include the output coordinate system, theme selections, output format and GIF pixel size. These common parameters are shown in the above figure enclosed within a dashed box.

These common parameters are required by the Servlet and must be present in any query web page file used by the Servlet. All web page files must include the HTML code that defines these parameters. The Servlet itself deals with them directly – the custom query page handler is not involved.

Any other parameters present in the query page are application-specific from the Servlet's standpoint and it is these custom parameters which are processed by the handler.

At a minimum a custom query page handler must be able to determine and return to the Servlet the bounds of the search rectangle defining the spatial extent of the query. Most if not all of a handler's custom web page query fields are used to obtain the information required to compute the search rectangle.

The determination of this rectangle can be very simple, as is the case with the default handler, which simply obtains the x and y coordinates of the rectangle's diagonally opposing corners directly from the form's entry fields.

However, search rectangle computation can also be complex, involving multiple methods for specifying the area of interest, each with its own set of computational parameters. The handler could even connect to an external database, perform queries to obtain additional information and use the results in the search rectangle computation.

To ensure that the common parameters required by the Servlet are always present, custom query web page files must be based on the default `fetchDefault.html` web page file that comes with the Servlet.

This default file should be copied and modified as necessary. It contains all the HTML code required to support the common parameters, as well as the code for the search area entry fields. The latter are the application-specific parameters for the default form, and it is their HTML code that should be replaced with custom code.

The code section that can be replaced is marked by comment lines in the file. It starts below the comment line `START OF DEFAULT CUSTOM QUERY PAGE HANDLER CODE` and ends above the comment line `END OF DEFAULT CUSTOM QUERY PAGE HANDLER CODE`. Code outside of this section should not be modified in any way.

Adding Custom Controls

Custom HTML input controls (such as text fields) can be also be added to DDE web pages. If these controls are within an HTML form that will be submitted to the Translation Servlet to initiate a translation request (such as the default order form), the values entered into these input controls can be passed on to the FME via the Servlet as macro definitions for use in customized mapping files. The details are as follows:

Each web page customized in this way must first specify (as an HTML hidden variable) a user-defined string that will be used as a common prefix in the HTML name of every site-specific input control in the web page. The HTML hidden variable specifying this prefix must be named `SpatialDirectParamPrefix`.

Once this prefix is specified in the web page, site-specific input controls can be added to it. The name given to each control must start with the prefix. The remainder of each control's name (after the prefix) will be used as the name of an FME macro whose value will be set to the value entered into the control.

An FME macro name-value definition thus constructed is created for every control in the web page whose name begins with the specified prefix. Each macro definition ultimately shows up in the FME translation command string generated by the Servlet and sent to the FME.

Associated site-specific modifications can then be made to the DDE mapping files to make use of the macro-defined values that originate as user-entered values in the custom web page input controls and are passed on to the FME via the Servlet using this control-name to macro-name mapping.

For example, an input text field representing road surface type could be added to the order form web page. If the user-defined prefix was chosen to be `FME_` then the web page would define this with:

```
<INPUT TYPE="HIDDEN" NAME="SpatialDirectParamPrefix" VALUE="FME_">
```

The input text field's name would consist of the prefix plus whatever macro name was descriptive of its value. If the macro name was chosen to be "SurfaceType" then the input text field's basic HTML definition would look similar to this:

```
<INPUT TYPE="TEXT" NAME="FME_SurfaceType" VALUE="">
```

If the user enters the string value "coarse gravel" into this field and submits the form, the Servlet sees that the control's name starts with the "FME_" prefix which identifies the control as one supplying a site-specific parameter value. The Servlet then uses the control's name suffix, in this case "SurfaceType", as a macro name whose value is set to be "coarse gravel". The command line sent by the Servlet to the FME will then include the macro definition:

```
--SurfaceType "coarse gravel"
```

In the DDE mapping file(s) there would be one or more references to the "SurfaceType" macro value used wherever the value is required, depending on the specific nature of the custom processing.

Query Page Instantiation

In the course of its operation, the Servlet sends dynamic-content web pages created on-the-fly. These are generated by the Servlet using a procedure called *dynamic instantiation*.

This procedure uses fixed HTML template files containing placeholder tokens within the file's static contents. At runtime, the Servlet replaces these tokens with the appropriate values and sends the resulting file to the browser. The values are obtained from a Properties hashtable called `userInfo`, whose keys are the names of the tokens and whose values are the token replacement values.

The handler's query web page file is one of these fixed HTML template files. Using Servlet dynamic instantiation, a handler can insert values into its web page file at runtime by putting the required key-value entries into the `userInfo` Properties hashtable. This hashtable is made available to the handler as an argument in the method calls to it.

Placeholder tokens within the HTML file must have the following format:

```
%tokenName%
```

where the token's name is enclosed within `%` characters. For example, the following HTML template file code:

```
<INPUT TYPE="TEXT" NAME="lowerLeftX" VALUE="%lowerLeftX%">
```

defines a text entry field called `lowerLeftX`. The value is set to a placeholder token named `lowerLeftX` as denoted by the enclosing percent signs. If the handler makes an entry in the `userInfo` Properties hashtable specifying a key of `lowerLeftX` and a value of `27.5`, the Servlet will replace the token with the value and send the following HTML code to the browser:

```
<INPUT TYPE="TEXT" NAME="lowerLeftX" VALUE="27.5">
```

If no `userInfo` entry is found for a token, the Servlet outputs the token as-is, including the percent signs, as a literal, without performing any replacement.

If a percent sign itself is needed as a literal, it should be entered twice. For example:

```
My score is %score%%%
```

would become after replacement:

```
My score is 32%
```

Note The requirement to express `%` sign literals as `%%` pairs means that all such literal occurrences within the HTML query page template file must be expressed in this way. For example, the HTML code to set a table width would be:

```
<TABLE WIDTH="95%%">
```

Note that the width of ninety-five percent is expressed as `"95%%"` rather than `"95%"`.

Query Page Handler Interface Methods

`IViewerQueryPageHandler` declares a number of methods that a custom query page handler must implement.

One of the arguments made available to these methods is `formParams`. This is a hashtable that holds the names and values of every HTML parameter in the form of the current web page being processed. In most cases this will be the query web page.

Most of the work of a query handler involves the transfer of query page state information to and from the Servlet before and after page instantiation. In general, the Servlet supplies the handler with existing page state contained in `formParams`, and the handler supplies the Servlet with desired page state contained in `userInfo`.

Note The Servlet uses multiple threads to process requests, but it dynamically loads, creates and uses a single instance of the query handler which is used by all threads. As a result, custom query handlers must have the following characteristics:

- Must be written so as to be thread-safe. Any instance variables that a handler uses must be assigned values at creation/initialization time only, and subsequent references to these variables must be read-only.
- Must supply a constructor method that takes no parameters.

The following sections describe each of the interface methods in detail.

Handler Construction

At start-up, the Servlet uses a default parameterless constructor to create a single instance of the handler that implements the `IViewerQueryPageHandler` interface.

Initialization

- **`public int init(Hashtable handlerInitParams)`**

  This method is called immediately after the Servlet creates the handler. It should perform whatever initialization is required by the handler.

  This usually involves saving the values of the initialization parameters handed to it in the `handlerInitParams` hashtable argument. These values can then be used by subsequent methods as needed. A handler can use none, some, or all of the supplied parameters.

Table 14-3 lists the keys and value descriptions of the parameters present in `handlerInitParams`:

TABLE 14-3 Handler Initialization Parameters

Hashtable Key	Parameter Type	Parameter Description
`viewer`	`ViewerServlet`	The Servlet itself, which allows the handler to communicate directly with the Servlet.
`supportDataDir`	String	The pathname of the Servlet’s support data directory, it allows the handler to access required support files.
`preferencesDir`	String	The pathname of the Servlet’s preferences directory which allows the handler to access the user properties files.
`urlBase`	String	URL of the `safeViewerHTML` directory on the web server.
`formatTypeTable`	Hashtable	Contains entries whose keys are mapping-filenames and whose values are format- types.
`formatTagTable`	Hashtable	Contains entries whose keys are mapping-filenames and whose values are format-tags.
`minXSearchLimit`	Double	Minimum x-coordinate of the source data.
`maxXSearchLimit`	Double	Maximum x-coordinate of the source data.
`minYSearchLimit`	Double	Minimum y-coordinate of the source data.
`maxYSearchLimit`	Double	Maximum y-coordinate of the source data.

This method should return `kVSSuccess` if initialization was successful and `kVSQueryPageHandlerInitFailed` if initialization failed. Both of these constants are defined in the interface called:

```
COM.safe.viewerservlet.IViewerServletConstants
```

Log-in

The following method is used in the Log-in procedure:

- **`public Properties ensureUserInfoCompleteness(Properties userInfo,Hashtable formParams,ArrayList coordSysList)`**

  This method is called when a user logs in via the log-in web page.

  The parameters `userInfo` are saved to the user’s preferences file by the Servlet between query sessions and are automatically present and available when `userInfo` is reloaded from the user's property file by the Servlet.

However, if these user parameters have never been saved before (newly added user, for example) they will not be available in `userInfo` until the first query takes place.

Some parameters may be required to be present in `userInfo` for the handler's query page to be correctly instantiated by the Servlet. These parameters would typically be placeholder tokens representing values in the query page that the handler uses. If there are any required parameters that are absent, this method should add them to `userInfo` with default values.

The Servlet first performs its own version of this method, which adds the user name and password to `userInfo`. The Servlet then invokes the handler's version of the same method.

The arguments given to the handler include `userInfo`, `formParams` which contains the log-in form parameter values and `coordSysList`.

The latter is an Array List reflecting the output coordinate system choices available in the query page's coordinate system pull-down list. `coordSysList` contains paired String elements. The first member of each pair is the FME coordinate system name and the second member is the associated coordinate system description.

This method should make any required additions to `userInfo` and return it.

Query Page Preparation

The following method is used in query page preparation:

- **`public Properties prepareForQueryPage(Properties userInfo)`**

  This method is called just before the Servlet creates and displays the query page. It is used by the handler to add any tokens or values to `userInfo` that are required for proper instantiation of its query page.

  Typical `userInfo` additions performed by this method are those associated with the handler's methods of query. For example, the handler might support query search rectangle specification by Lat./Long., township number and NTS block ID. Each of these query methods could have associated parameter tokens which would be candidates for entry into `userInfo`.

  When `userInfo` is passed to this method, it will contain the minimum and maximum x and y coordinates of the current search rectangle, as long as at least one translation has been performed. The handler can test for the presence of the keys `minX, maxX, minY` and `maxY` in `userInfo` and use their values if desired.

  This method is given `userInfo` to update as required and should return `userInfo` to the Servlet.

Query Processing

The following methods are called to process the query when the user has completed query entry and submits the query page form. The `formParams` argument supplied to some of the methods contains the names and values of all the form parameters in the handler's query page when the user submitted the form.

- **`public BoundsBox getSearchRect(HttpServletResponse response,Hashtable formParams)`**

  This method is called to determine the bounds of the query search area as a rectangle and return them as a `COM.safe.utility.BoundsBox`. The constructor for a `BoundsBox` is:

  ```
  BoundsBox(double minX,double maxX,double minY,double maxY)
  ```

  where the parameters represent the x and y coordinates of the search rectangle's diagonally opposing corners.

  This method is called by the Servlet when `usesSearchPolygon()` returns `false`.

  If the search area cannot be determined, the method should return `null`.

  For arguments the method is given the JSDK `HttpServletResponse` object associated with the query request and `formParams`.

- **`public boolean usesSearchPolygon()`**

  This method is called by the Servlet to determine if the query handler can return the query search area as a general, multi-vertex polygon (return `true`) or strictly as a rectangle (return `false`).

Note The current version of the Servlet accepts only rectangular search areas. As a result, all query handlers should implement this method to return false., causing the Servlet to always call getSearchRect().

- **`public ArrayList getSearchPolygon(HttpServletResponse response,Hashtable formParams)`**

  This method is called to determine the bounds of the query search area as a general, multi-vertex polygon. It is called by the Servlet when `usesSearchPolygon()` returns `true`.

  The search area polygon should be returned as an ArrayList containing one `COM.safe.utility.ThreeDPosition` object for every vertex. The constructor for a `ThreeDPosition` is:

  ```
  ThreeDPosition(double xCoord, double yCoord, double zCoord)
  ```

  where the parameters represent the x, y and z coordinates of the vertex.

  `ThreeDPositions` should be stored in the ArrayList in counter-clockwise order around the polygon. The final position in the ArrayList must be identical to the first one in order to form polygon closure. This means that for a polygon containing N vertices, the ArrayList will contain N+1 `ThreeDPositions`.

If the search area cannot be determined, the method should return `null`.

As arguments the method is given the JSDK `HttpServletResponse` object associated with the query request and `formParams`.

- **`public Properties saveQueryPageState(Hashtable formParams,Properties userInfo)`**

  This method should add to `userInfo` any tokens/values associated with the query page that need to be saved and reproduced the next time the page is instantiated. Entries to `userInfo` will be stored and subsequently used to do the instantiation.

  Typical `userInfo` additions performed in this regard are those associated with the search area bounds. For example, the default handler obtains from `formParams` the values of the (x,y) coordinates of its search rectangle's diagonally opposite corners and saves them into `userInfo`, causing them to be appear in the page when it is next instantiated. This allows the page to "remember" the user's last entered search bounds.

  This method is given `userInfo` to update as required as well as `formParams` and should return `userInfo` to the Servlet.

- **`public String getCustomCommandSubstring()`**

  This method can be used to return a custom substring for inclusion in the FME command line that is built by the Servlet. This substring can have any arbitrary content. If no custom substring is desired, this method should return a blank string in the form of " " – two double-quote characters.

  The Servlet doesn't check, test or interpret the substring returned by this method. It simply adds the substring to the FME command string it builds and sends the result to the FME. The content of this substring must make sense to the FME.

  The Servlet also propagates this substring through to its zoom/pan and refetch operations.

  Note that the Servlet includes this substring only when the interactive query form page is used to supply values for the query. The substring is not included when remote fetches are performed. However, the latter can cause an equivalent substring to be included by specifying the `fmeParams` parameter.

HTML Template Placeholders

Much of the Servlet's activity involves communicating with a browser. It does this by sending HTML web pages to the browser, then responding to user-generated events on those pages from the browser. There is a tight logical coupling between the Servlet and the HTML pages it uses.

In the course of its operation, the Servlet sends both fixed-content and dynamic-content pages. The latter are generated using a procedure called *dynamic instantiation*. This procedure uses fixed HTML template files containing placeholder tokens within the file's static contents. At run-time, the Servlet replaces these placeholders with the appropriate values and sends the resulting file to the browser.

The Servlet uses these dynamically created pages to maintain state information across request-response page changes on the browser. Much of this information is stored in HTML variables whose assigned values are among those determined during page instantiation.

The following sections describe the placeholders present in the most commonly used HTML template files.

General-Use Placeholders

A number of general-use placeholders are used by several template files. These placeholders are:

fmeLog

The name of the FME translation log file located in the translation results directory. Determined by the FME server.

outputFormatOptions

The HTML code defining the contents of the "Format" pull-down list on the query order form web page, and the "Download as:" pull-down list on the GIF image map results web page. Determined by the Servlet from information in the `outputFormats.csv` file.

resultsTempDir

The directory portion of the URL of the translation results directory. Determined by the FME server.

servletURL

The URL of the Servlet. Set in the Servlet's `servlets.properties` file.

urlBase

The URL of the safeViewerHTML directory located in the web server's public root directory. Set in the Servlet's `servlets.properties` file.

Specific-Use Placeholders

Several template files use placeholders that are specific to their function. The following sections describe these files and their specific placeholders.

fetchDefault.html

The `fetchDefault.html` template file defines DDE's default query order form page. It uses the following placeholders:

- **`checked`**

  The `CHECKED` toggle value to cause the "enableAutoThemeSelection" check-box to be checked or not. Set by the user and maintained by the Servlet.

- **`dbUnitsChecked`**

  The `CHECKED` toggle value to cause the default "dbUnits" search area method radio button to always be selected. Set by the Servlet.

- **`layerListCode`**

  The HTML code defining the theme pull-down lists. Determined by the Servlet from information in the `config.csv` file.

- **`lowerLeftX, upperRightX, lowerLeftY, upperRightY`**

  The minimum and maximum x and y coordinates of the current query search rectangle. These values appear in the page's "Search Area:" text entry fields. Set by the user and maintained by the Servlet.

- **`minXSearchLimit, maxXSearchLimit, minYSearchLimit, maxYSearchLimit`**

  The minimum and maximum x and y coordinates of the total extent of the source data. Set in the Servlet's `servlets.properties` file.

- **`outputCoordSysOptions`**

  The HTML code defining the contents of the "Coordinates:" pulldown list on the query order form web page. Determined by the Servlet from information in the `outputCoordSystems.csv` file.

fetchGifSuccess.html

The `fetchGifSuccess.html` template file defines DDE's GIF image map results page. It uses the following placeholders:

- **`coordsys`**

  The coordinate system of the translated output. This is the FME name of the system, such as `BCALB-83`. Determined by the Servlet from information in the "Coordinates:" pulldown list on the query order form web page, or a remote fetch URL.

- **`enableAutoThemeSelection`**

  The switch enabling/disabling automatic theme selection by the Servlet. Indicates whether or not this was in effect for the current translation result. Set by the user and maintained by the Servlet.

- **`gifResult`**

  The name of the GIF image file resulting from the current translation. Determined by the FME server.

- **`imageMap`**

  The HTML code defining the image map associated with the GIF image. Generated by the GIF Writer and inserted by the Servlet.

- **`legend`**

  The HTML code defining the legend associated with the GIF image. Generated by the GIF Writer and inserted by the Servlet.

- **`mappingFilename`**

  The name of the FME mapping file used in the translation. Determined by the Servlet from information in the "Format" pulldown list on the query order form web page, or the "Download as:" pulldown list on the GIF image map results web page, or a remote fetch URL.

- **`minX, maxX, minY, maxY`**

  The minimum and maximum x and y coordinates of the current query search rectangle. Set by the user and maintained by the Servlet.

- **`panFactorPercentIndex`**

  The zero-based index into the array of pan percentage factors available in the "Pan %" pulldown list. This value indicates which pan percentage was in effect for the current translation result. Set by the user and maintained by the Servlet.

- **`panSelect0...n`**

  The `SELECTED` toggle value to cause one of `n` pan percentages in the "Pan %" pulldown list to be selected or not. Set by the user and maintained by the Servlet.

- **`pixels`**

  The height and width of the GIF image (when generated) in pixels. Determined by the Servlet from information in the "Pixels:" pulldown list on the query order form web page, or a remote fetch URL.

- **`scale`**

  Currently unused. Reserved for future use.

- **`translationSubCommand0...n`**

  The constant portion of the FME translation command string that generated the GIF image. This portion does not change when pans, zooms or non-GIF translations are initiated from this page. There are `n translationSubCommand` placeholders, distinguished from one another by a trailing digit. The translation subcommand is spread across as many of these `n` placeholders as required when its length exceeds the maximum allowed for the value of a single HTML variable. Set by the Servlet.

- **`userSelectedThemes0...n`**

  The list of user-selected themes displayed in the GIF image. There are `n userSelectedThemes` placeholders, distinguished from one another by a trailing digit. The theme list is spread across as many of these `n` placeholders as required when its length exceeds the maximum allowed for the value of a single HTML variable. Set by the Servlet.

- **`zoomFactorIndex`**

  The zero-based index into the array of zoom factors available in the zoom "Amount" pulldown list. This value indicates which zoom magnification was in effect for the translation result. Set by the user and maintained by the Servlet.

- **`zoomSelect0...n`**

  The `SELECTED` toggle value to cause one of `n` zoom magnifications in the zoom "Amount" pulldown list to be selected or not. Set by the user and maintained by the Servlet.

fetchZipSuccess.html

The `fetchZipSuccess.html` template file defines DDE's non-image results page. Non-image translation results are zip-compressed and downloaded from this page. It uses the following placeholders:

- **`resultsDeletionTime`**

  The datetime stamp defining the time after which the translation result will be deleted by the FME server.

- **`zipResult`**

  The name of the compressed zip file containing the results of the current translation. Determined by the FME server.

Servlet Properties

Servlet properties can be modified to change the behaviour of the Servlet. Where and how these modifications are made depends on the servlet engine in use.

Servlet Engines

The default DDE configuration uses one of two possible servlet engines to run the Servlet – Tomcat or ServletRunner – both of which are bundled with the DDE distribution. Each of these engines obtains servlet property values at start-up time from an engine-specific parameter file.

One of these engines (depending on the platform) is always installed automatically as part of the DDE installation and is used solely by DDE to run the Translation Servlet. DDE's Tomcat and ServletRunner engines are separate from and independent of any other servlet engine that may be present on the same system.

Request Forwarding

Translation requests sent to DDE's servlet engine (Tomcat or ServletRunner) are routed to the engine indirectly through the system webserver's public request port (default 80).

This routing is performed by a CGI Perl script called `spatialDirect.pl` which is copied during installation into a user-specified subdirectory within the webserver's main CGI script directory.

At runtime a translation request is sent like any other public request to the webserver at (by default) port 80. The request includes a reference to the `spatialDirect.pl` Perl script, causing the latter to process the request. This processing consists of forwarding the request on to the DDE servlet engine which is listening on its own port (8194 by default). The latter then hands the request to the Translation Servlet as usual.

The same `spatialDirect.pl` Perl script waits for a response from the Translation Servlet and sends it back to the client that originated the request.

Please note that translation requests can still be sent directly to the servlet engine port rather than to the webserver public port for forwarding. However, the forwarding mechanism allows translation requests to be received by DDE when the latter is operating behind a firewall, and the webserver's public port is the only one available to accept requests.

Tomcat Servlet Engine

DDE for Windows and Linux uses the Jakarta Tomcat servlet engine to run the Servlet. Configuration properties for the Servlet are stored in the Tomcat `web.xml` file.

This file is used to set the configuration properties of the Servlet. Property names and values are specified using XML tag syntax.

In the default DDE configuration, this Tomcat properties file is located in:

```
<DDEInstallDir>/tomcat/webapps/servlet/WEB-INF/web.xml
```

Note: The sections below assume the use of the ServletRunner engine when describing Servlet properties. Although Tomcat property names and values are specified using XML tag syntax, the property names and values themselves are identical to those used with the ServletRunner (or any other) servlet engine.

ServletRunner Servlet Engine

DDE for all other platforms uses Sun's ServletRunner to run the Servlet. Configuration or properties for the Servlet are stored in the ServletRunner's `servlets.properties` file.

This file is used to set the configuration properties of the Servlet. In the default DDE configuration, this properties file is located in:

```
   <DDEInstallDir>/Jsdk/webpages/WEB-INF/servlets.properties
```

Other Servlet Engines

The Servlet can be run using any standard servlet engine (or servlet-enabled web server) and is not restricted to running under Tomcat or ServletRunner. For further details, see Chapter 16, *Servlet Engine Configuration*.

Property Modification

Modification of the Servlet's property values can be performed in order to customize the Servlet's behavior for site-specific conditions. Although Servlet properties in the properties file can be manually edited with any text editor, it is usually more convenient to use the Property Editor utility instead.

When modifying property values, the following points should be noted. Note that the sections below assume the use of the ServletRunner engine when describing Servlet properties.

- The `<servletrunner-port>` number defined in the `servletURL` property must match the port number assigned to the `server.port` parameter in the ServletRunner's `default.cfg` file.
- The `<fme-server-URL>` defined by the `layerListConfURL` property is the URL pointing to the location on the FME Server of the `config.csv` file. This file is required if the default DDE configuration is used.
- The `<QServer-hostname>` defined by the `serviceManagerHost` property is the name of the host on which the QServer is running, if it's in use. If it isn't in use, no value needs to be entered.
- The `<QServer-request-port>` defined by the `serviceManagerRequestPort` property is the port number on the QServer system on which the QServer listens for translation requests. The port number entered here must match the port number assigned to the `REQUEST_PORT` parameter in the QServer's configuration file. If the QServer is not in use, no value needs to be entered.
- Path names should use the forward-slash character (/) as the separating character. Path names should **not** be enclosed in quotations, even if they contain spaces. The Servlet will automatically enclose space-containing path names, as needed.

  For example, the following is a correctly formed Servlet property pathname:

  ```
  C:/Program Files/FME/fme
  ```
- Any `#` character at the beginning of a line marks a comment line which is ignored by the ServletRunner.
- Multiple name-value pairs are separated by commas.
- The `\` character at the end of a physical line acts as a line continuation character.
- Order of parameter appearance in the file is unimportant.

Note: When Servlet property values are modified, the servlet engine (Tomcat or ServletRunner) must be restarted for the new values to take effect. This usually requires restarting DDE.

Property Descriptions

This section describes the Servlet properties grouped according to function.

Translation Properties

- **`preferencesDir`**

  Represents the full pathname of the directory containing the user preferences files, `.prp` extension, which store various user-specific parameters such as user name, e-mail address and selected layer list options.

  Example: `C:/Program Files/DDE/safeViewerHTML/prefs`

- **`supportDataDir`**

  Represents the full pathname of the directory containing site-specific data files to support custom queries, search area computations, etc.

  Example: `C:/Program Files/DDE/safeViewerHTML/supportData`

Security Properties

- **`prohibitedTranslationParams`**

  This property specifies a list of parameter names that are not allowed to be present in the optional `fmeParams` value that is appended (if present) to the translation command sent to the FME Server. The property value takes the form of a pipe-delimited string with each field representing a string that is prohibited from being in the `fmeParams` value.

  The Servlet removes all prohibited strings that it finds in the `fmeParams` value before passing the latter on to the FME Server. The Servlet also logs a warning for each prohibited string it removes. Note that removal of these strings can result in a malformed translation command, causing the FME Server to abort the translation.

  The standard DDE distribution sets this property to a list of default prohibited strings which should be suitable as-is for the majority of sites. Users can modify this list as required to suit the specific needs of their environment.

Note: If this property is absent (as can occur when using an older properties file with a newer version of DDE) the Servlet logs a warning but will continue to operate. However, it is strongly recommended that this property always be present and defined with at least the list of default prohibited strings. Users should contact Safe Software to obtain the default list if they do not already have it.

Example: `numLanes|pavementType|medianPresent`

HTML Properties

- **`cookieDomain`**

  Represents the domain that will be assigned to any cookies sent by the Servlet to the browser. The domain value must begin with a dot character and contain at least two dots. To have the system default be used for this property, assign a value of `default` (`cookieDomain=default,\`) meaning that the domain is the host that created the cookie.

  Example: `.noodle.com`

- **`cookiePath`**

  Represents the pathname that will be assigned to any cookies sent by the Servlet to the browser. The most general and widest scope path to assign to a cookie to is `/`—the root directory. To have the system default be used for this property, assign a value of `default` (`cookiePath=default,\`) meaning that the path points to and below where the cookie was defined.

  Example: `/`

- **`fileLoadURLBase`**

  Represents an alternate root URL address location where the Servlet should go to obtain HTML code required for the default map results display. In most cases this property need not be specified, and the Servlet will use the `resultPrefix + resultRootDir` default location returned by the FME Server to obtain the required code. However, there may be cases (usually involving firewalls) where this location cannot be accessed by the Servlet. If this occurs the map results display will have no legend and no popup attribute panels. In these cases the `fileLoadURLBase` property can be specified to provide an alternate URL for the Servlet to use in obtaining the HTML code. The URL is typically a `file://` local filesystem reference to the translation results directory.

  Example: `file://c:/Program Files/apache/htdocs/results`

- **`htmlTemplatesDir`**

  Represents the full pathname of the directory containing all Servlet HTML template source files. These are `.html` files which are dynamically modified whenever they are downloaded to the browser to reflect parameters specific to the user's current translation session.

  The `.html` files in this directory are not directly viewed in a web browser, but rather are instantiated by the Servlet when they are sent to the browser. At instantiation time, any dynamic parameters they reference are replaced by the settings particular to the user.

  Example: `c:/htdocs/safeViewerHTML/htmlTemplates`

- **`servletURL`**

  Represents the URL address of the ServletRunner and ServletRunner name of the Servlet servlet.

  Example: `http://JIM:8194/servlet/translationServlet`

  In this example the ServletRunner would need to be running on host `JIM` and listening on port 8194. “`translationServlet`” is the name of the Servlet that the ServletRunner should run and would be defined in its `servlets.properties` file.

Note: The port number specified here must match the port number assigned to the "`server.port`" parameter in the ServletRunner’s `default.cfg` file.

- **`urlBase`**

  Represents the root URL address location where web-related Servlet files and subdirectories are stored. This includes `.html` source files, image files and user preference files. This location is usually a subdirectory located within the public HTML root directory of the system’s web server.

  Example: `http://JIM:80/safeViewerHTML`

Order Form Properties

- **`defaultNotificationEmailAddress`**

  Represents the default e-mail address to send asynchronous result notifications to if no such address is otherwise specified.

- **`layerListConfURL`**

  Represents the URL pointing to the location on the FME server of the `config.csv` file.

  Example: `ftp://WORF/fme/translationControl/system/config.csv`

- **`layerListHeight`**

  Represents the integer number of choices displayed before having to scroll each selection theme list of the order form page. If this property is set to a value which cannot be interpreted as an integer, the Servlet will use a default value of `5` instead. Increasing the value results in a order form which will occupy more vertical space on the web page.

  Example: `5`

- **`maxNumLayerListColumns`**

  Represents the maximum integer number of columns to display in each selection theme list row of the order form page. If this property is set to a value which cannot be interpreted as an integer, the Servlet will use a default value of 2 instead.

Example: `2`

- **`notificationEmailSubjectLine`**

  Represents the subject line text of translation result notification e-mail.

  Example: `SpatialDirect Result Notification`

- **`numSubCommandSegments`**

  Represents the number of HTML hidden variables in the `fetchGifSuccess.html` page file that are available to store portions of the FME command string. The file as distributed supplies ten variables which should be sufficient for the great majority of cases. This is the value assigned in the `servlets.properties` file and should not require modification.

  Additional hidden variables are required only if the portion of the FME command string that is saved in them exceeds (`1024 * numSubCommandSegments`) characters in length. Additional variables can be simply added to the existing list in the `fetchGifSuccess.html` file and the new total number set as the new value for the `numSubCommandSegments` property.

  Example: `10`

- **`numUserSelectedThemeSegments`**

  Represents the number of HTML hidden variables in the `fetchGifSuccess.html` page file that are available to store portions of the user-selected theme string. The file as distributed supplies ten variables which should be sufficient for the great majority of cases. This is the value assigned in the `servlets.properties` file and should not require modification.

  Additional hidden variables are required only if the portion of the user-selected theme string that is saved in them exceeds (`1024 * numUserSelectedThemeSegments`) characters in length. Additional variables can be simply added to the existing list in the `fetchGifSuccess.html` file and the new total number set as the new value for the `numUserSelectedThemeSegments` property.

  Example: `10`

- **`outputCoordSysPathName`**

  Represents the full pathname of the `outputCoordSystems.csv` file. This is a comma-separated-value ASCII text file containing entries specifying which output coordinate systems will appear as choices in the Servlet order form's `Coordinates` pull-down list. The pathname value assigned to this property must be of the form:

  ```
  <DDEInstallDir>/safeViewerHTML/supportData/
  outputCoordSystems.csv
  ```

  Example: `c:/Program Files/DDE/safeViewerHTML/supportData/outputCoordSystems.csv`

- **`outputFormatsPathName`**

  Represents the full pathname of the `outputFormats.csv` file. This is a comma-separated-value ASCII text file containing entries specifying which output formats will appear as choices in the Servlet order form's `Format` pull-down list.

  The pathname value assigned to this property must be of the form:

  ```
  <DDEInstallDir>/safeViewerHTML/supportData/
  outputFormats.csv
  ```

  Example: `c:/Program Files/DDE/safeViewerHTML/supportData/outputFormats.csv`

- **`queryPageHandlerClassName`**

  Represents the fully-qualified Java class name of the class associated with the order form page named by the `queryPageHandlerHTMLFile` property (see next property). The class must be accessible via the Java CLASSPATH environment variable.

  This handler class is dynamically loaded by the Servlet at start-up time and is used by the Servlet to handle the exchange of parameters specific to the named order form page.

  Example: `COM.safe.viewerservlet.DefaultQueryPageHandler`

  The example above specifies the default handler supplied by the Servlet, which is suitable in most cases for basic, general-purpose use.

- **`queryPageHandlerHTMLFile`**

  Represents the name and extension of the HTML file that defines the Servlet's order form page. This file must be located in `<webserver-html-root-dir>\safeViewerHTML\htmlTemplates`.

  The Servlet uses the order page handler named by the `queryPageHandlerClassName` property (see previous property) to access parameters on this page.

  Example: `fetchDefault.html`

  The example above specifies the default order form page supplied by the Servlet, which is suitable in most cases for basic, general-purpose use.

- **`requestFilterClassName`**

  Represents the fully-qualified Java class name of the class associated with the request filter. The class must be accessible via the Java CLASSPATH environment variable.

  This filter class is dynamically loaded by the Servlet at start-up time and is used by the Servlet to determine whether or not a request should be processed immediately or delayed until a later time.

  The request filter also supplies a request complexity value which is used by the Servlet to compute request cost when translation purchasing is in effect.

Example: `COM.safe.viewerservlet.DefaultRequestFilter`

The example above specifies the default filter supplied by the Servlet, which is suitable in most cases for basic, general-purpose use.

- **`requestFilterParam`**

  Represents the parameter string used by the request filter.

  Example: `14.0|8:00|18:00`

  The example above specifies a parameter string used by the default request filter supplied by the Servlet. The first value represents the request complexity limit, and the remaining value pair defines the hours of the day of the off-peak period.

- **`smtpHost`**

  Represents the hostname of the system running the SMTP mail protocol which should be used to send translation result notification e-mail.

  Example: `spock`

- **`smtpSenderAddress`**

  Represents the e-mail address on the SMTP host which should be used as the sender of translation result notification e-mail.

  Example: `sender@mailhost.com`

- **`sortAscending`**

  Represents a true-or-false string specifying whether or not the Servlet should sort the theme names as they appear in each selection theme list in the order form page in alphabetically ascending or descending order. The value assigned to this property is used only if the `sortSelectOptions` property is `true`. The value should be specified as either `true` or `false`. If any other value is specified, the servlet will use a default value of `true`.

  Example: `true`

- **`sortSelectOptions`**

  Represents a true-or-false string specifying whether or not the Servlet should alphabetically sort the theme names as they appear in each selection theme list in the order form page. The value should be specified as either `true` or `false`. If any other value is specified, the servlet will use a default value of `true`.

  Example: `true`

- **`useRequestFilter`**

  Represents a true-or-false string specifying whether or not the Servlet should perform request filtering. The value should be specified as either `true` or `false`. If any other value is specified, the servlet will use a default value of `false`.

Example: `false`

Search Properties

- **`maxXSearchLimit`**

  Represents the floating-point value of the query search rectangle's maximum allowable x-coordinate. This limit is enforced when users enter values into the order form page and when they are zooming or panning.

  If this property is set to a value which cannot be interpreted as a floating-point number, the Servlet will use a default value of `1000000.0` (one million) instead, **and** set the other three remaining search limit property values to their default values as well.

  Example: `120000.0`

- **`maxYSearchLimit`**

  Represents the floating-point value of the query search rectangle's maximum allowable y-coordinate. This limit is enforced when users enter values into the order form page and when they are zooming or panning.

  If this property is set to a value which cannot be interpreted as a floating-point number, the Servlet will use a default value of `1000000.0` (one million) instead, **and** set the other three remaining search limit property values to their default values as well.

  Example: `60.0`

- **`minXSearchLimit`**

  Represents the floating-point value of the query search rectangle's minimum allowable x-coordinate. This limit is enforced when users enter values into the order form page and when they are zooming or panning.

  If this property is set to a value which cannot be interpreted as a floating-point number, the Servlet will use a default value of `0.0` instead, **and** set the other three remaining search limit property values to their default values as well.

  Example: `27000.0`

- **`minYSearchLimit`**

  Represents the floating-point value of the query search rectangle's minimum allowable y-coordinate. This limit is enforced when users enter values into the order form page and when they are zooming or panning.

  If this property is set to a value which cannot be interpreted as a floating-point number, the Servlet will use a default value of `0.0` instead, **and** set the other three remaining search limit property values to their default values as well.

Example: `20.0`

QServer Properties

- **`serviceManagerHost`**

  Represents the name of the host system on which the QServer is running and to which the Servlet will send FME server translation requests.

  Example: `JIM`

- **`serviceManagerRequestPort`**

  Represents the network port number on the `<serviceManagerHost>` system on which the QServer is listening for FME translation requests and to which the Servlet will send such requests.

  Example: `7071`

- **`useServiceManager`**

  Represents a true-or-false string specifying whether the Servlet should use the QServer to forward translation requests to FME servers, or perform translations with a locally-run FME instead.

  This property should always be set to `true` for the Servlet to use the QServer. This is the default value.

  Example: `true`

Logging Properties

- **`appendToExistingLog`**

  Represents a true-or-false string specifying whether or not the Servlet should append its message log output to the existing log file (`true`) or overwrite the existing log file (`false`), destroying its previous contents. The value should be specified as either `true` or `false`. If any other value is specified, the Servlet will use a default value of `false`, overwriting the existing file.

  Example: `true`

- **`echoLogToConsole`**

  Represents a true-or-false string specifying whether or not the Servlet should send its message log output to the console in addition to the log file. The value should be specified as either `true` or `false`. If any other value is specified, the servlet will use a default value of `false`.

  Example: `true`

- **`enableRequestLogging`**

  Represents a true-or-false string specifying whether or not the Servlet should include additional information about the incoming translation request in its message log output. The extra information specifies the following:

  - requesting host's name and address
  - request scheme and protocol
  - request method and URL
  - request parameters
  - request headers

  The value of this property should be specified as either `true` or `false`. If any other value is specified, the Servlet will use a default value of `false`.

  Example: `true`

- **`includeThreadnameInLog`**

  Represents a true-or-false string specifying whether or not the Servlet should include the name of the current execution thread in its message log output. The value should be specified as either `true` or `false`. If any other value is specified, the Servlet will use a default value of `false`.

  Example: `true`

- **`includeTimestampInLog`**

  Represents a true-or-false string specifying whether or not the Servlet should include a timestamp in its message log output. The value should be specified as either `true` or `false`. If any other value is specified, the Servlet will use a default value of `false`.

  Example: `true`

- **`logPathName`**

  Represents the full pathname of the file used to contain all log messages generated by the Servlet. Such messages include informational, warning and error descriptions created during the course of Servlet operations.

  If the log file already exists, messages are appended to it or not, depending on the value assigned to `<appendToExistingLog>`. If the log file does not exist, the first message logged to it causes it to be created. This log file is different from and independent of the log files generated by the QServer, ProcessMonitor and FME.

  Example: `c:/Temp/ViewerServlet.log`

- **`maxLogfileAgeSeconds`**

  The maximum allowable age in seconds of previous versions of logfiles. Any logfiles older than this are deleted. Deletion of older logfiles only occurs when the current logfile exceeds `MAX_LOGFILE_LINES` in size and is closed. Optional.

Values < 0 mean previous logfiles are NEVER deleted. A value = 0 means that ALL previous logfiles are deleted. Values > 0 mean previous logfiles older than the specified value are deleted.

If absent, this parameter is assigned a default value of 604800 seconds (7 days).

Example: `100000`

- **`maxLogfileLines`**

  The maximum number of lines written to the current logfile, after which it is closed, followed by possible deletion of older ones and creation of a new one to continue on with. Optional.

  Values <= 0 mean size-limiting is NOT in effect. The same log file is used and can grow indefinitely if logfile appending is in effect. Values > 0 mean size-limiting IS in effect using the number of lines specified.

  If absent, this parameter is assigned a default value of 3000 lines.

  Example: `3000`

Miscellaneous Properties

- **`enableAutoThemeSelection`**

  Represents a true-or-false string specifying whether or not the Servlet should automatically select themes based on the current search area extent.

  This property determines the initial setting (enabled or disabled) for new users. Subsequent settings are determined by the user's adjustment of a control in the order page.

  The themes that are actually selected can be adjusted using the `Min Visibility Area` and `Max Visibility Area` fields in the `config.csv` file. Auto-selected themes are added to any user-selected themes that may be present.

  The value should be specified as either `true` or `false`. If any other value is specified, the Servlet will use a default value of `false`.

  Example: `true`

Synchronous Result Return

This section describes synchronous translation result returns by the Servlet. A *synchronous* return is one that the client waits for after making the request.[1]

When a translation request has been completed, the Servlet returns the result to the waiting client in the form of an HTML web page. Strictly speaking, the Servlet returns the stream of characters that constitutes the HTML code that defines the result web page which is displayable as such when processed by an HTML-aware client such as a browser.

However, there may be cases where the client is not a browser and has no interest in interpreting the HTML code as a displayable page. Rather, the client may wish to extract only the information describing the result from the HTML character stream, and ignore all page display information.

To support this extraction of result information from the HTML return stream, the Servlet inserts the information into the stream in the form of an HTML comment. This renders the information invisible to standard web browsers, while making it available to other client programs.

Result Responses

The HTML comment containing the result information can be one of several response types, depending on the nature of the returned data.

- If an FME translation result is available, the `TRANSLATION_RESULT_MSG` response type is returned.
- If no result is available (for example, there are no features returned, result inaccessible, translation exception, etc.) the `FIXED_RESULT_MSG` response type is returned instead.

Response Delimiters

To aid in extracting the response, the HTML result comment always starts with the delimiter string `<!-- RESULTSTART` and ends with the delimiter string `RESULTEND -->`. Within these two delimiters is the result response whose fields contain the result information. Some of these fields are composite, and contain their own subfields, while others contain just a single value. The fields are separated from each other by delimiter characters and are described in the sections below.

1. Conversely, *asynchronous* returns are those that the client does not wait for. Instead, e-mail notification is sent to the user once results are available. Asynchronous returns are described later in this document.

Response Syntax

The syntaxes of the various types of result response are described in the following sections.

`TRANSLATION_RESULT_MSG` Response Syntax

The `TRANSLATION_RESULT_MSG` response type is returned when a translation result is available. (This includes results from failed FME translations when a result is available describing the failure). This response type has the following general syntax:

```
<!-- RESULTSTART^TRANSLATION_RESULT_MSG^QServer
response^result file URL^RESULTEND -->
```

The response contains three fields, delimited by the “^” (caret) character.The first field is the fixed `TRANSLATION_RESULT_MSG` keyword identifying the response type. The second field is a composite one, and holds the QServer component of the response. The third field holds the URL pointing to the actual result file. This field is often the one of greatest interest to a client.

The QServer and result URL fields are described in the following sections.

QServer response field

The QServer response field is a composite field holding the QServer component of the response. It is composed of the following fields delimited by the “?” character:

- **result id**

  The content of the result id field takes the form:

  `id=`*`n`*

- **FME Server response**

  The FME Server response field is also a composite field holding the FME Server component of the response. It is composed of the following fields delimited by the “|” pipe character:

`num features output` (absent if translation failure)	The content of the num features output field takes the form: `NumFeaturesOutput=`*`n`*
`result file pathname` (absent if translation failure)	The content of the result file pathname field takes the form: `ResultDataset=`*`resultPathname`*
`result FME translation logfile pathname`	The content of this field takes the form: `ResultLog=`*`logPathname`*

`result lifetime in seconds before deletion` (absent if translation failure)	The content of this field takes the form: `ResultLifetime=`*`n`*
`result message`	The content of the result message field takes the form: `result=0:Translation Successful` | *`errorNum:errorMsg`*
`result root directory URL`	The content of the result root directory URL field takes the form: `ResultRootDir=`*`rootDirURL`*
`result URL protocol prefix`	The content of the result URL protocol prefix field takes the form: `ResultPrefix=protocol`

- **result success flag**

  The content of the result success flag field takes the form:

  ```
  resultSuccess=true | false
  ```

- **service message**

  The content of the service message field takes the form:

  `serviceMsg=`*`serviceNum:serviceMsg`*

- **service success flag**

  The content of the service success flag field takes the form:

  ```
  serviceSuccess=true | false
  ```

- **time requested**

  The content of the time requested field takes the form:

  `timeRequested=`*`timestamp`*

- **time started**

  The content of the time started field takes the form:

  `timeStarted=`*`timestamp`*

- **time finished**

  The content of the time finished field takes the form:

  `timeFinished=`*`timestamp`*

- **requester result port**

  The content of the requester result port field takes the form:

  `requesterResultPort=`*`n`*

- **requester host**

  The content of the requester host field takes the form:

  ```
  requesterHost=hostname | address
  ```

- **FME translation command**

  The content of the FME translation command field takes the form:

  ```
  request=command
  ```

- **request FME Server subsection keyword**

  The content of this field takes the form:

  ```
  requestKeyword=SAFE_VIEWER_DISPLAY_RESULT |
            SAFE_VIEWER_NONDISPLAY_RESULT
  ```

- **request priority**

  The content of the request priority field takes the form:

  ```
  priority=n
  ```

- **request description**

  The content of the request description field takes the form:

  ```
  description=description
  ```

Result File URL Field

The result file URL field contains the URL pointing to the actual translation result. This can be a displayable image file such as a GIF or a compressed ZIP file containing the translation output file(s).

`FIXED_RESULT_MSG Response Syntax`

The `FIXED_RESULT_MSG` response type is returned when a translation result is not available. This includes cases of no features returned, result inaccessible, translation exception, etc. The response has the following general syntax:

```
<!-- RESULTSTART^FIXED_RESULT_MSG^fixed result message^RESULTEND -->
```

The response contains two fields, delimited by the “^” character.The first field is the fixed `FIXED_RESULT_MSG` keyword identifying the response type. The second field is the actual fixed result message string and is generally an error message of some kind describing why a translation result is unavailable.

Files Returning Result Responses

A number of DDE HTML files return a result response. Client programs can always determine if a result response is available in the HTML return stream by searching for the presence of the `<!-- RESULTSTART` and `RESULTEND -->` delimiter strings. Each of the following standard DDE files returns a result response of the type shown:

HTML File	Type of Result Response
`fetchException.html`	`FIXED_RESULT_MSG`
`fetchFailure.html`	`TRANSLATION_RESULT_MSG`
`fetchGifSuccess.html`	`TRANSLATION_RESULT_MSG`
`fetchZipFailure.html`	`FIXED_RESULT_MSG`
`fetchZipSuccess.html`	`TRANSLATION_RESULT_MSG`
`outputNonExistent.html`	`FIXED_RESULT_MSG`
`outputNotAccessible.html`	`FIXED_RESULT_MSG`

Including User Information in Result Responses

A user-specified information string can be included in the result response web pages sent back to the client by DDE.

The FME Server config file is used to pass this user information back via the result web page. It's important to note that the FME Server reads its config file only once at start-up; therefore, the user information is set at FME Server start-up time, and cannot be dynamically changed on a per-translation basis. The FME Server must be restarted to make user information changes take effect. The FME Server's configuration pseudo-variables are also available for inclusion within the user information string.

This method enables user-defined strings to appear within the HTML comment string located at the beginning of the DDE result web pages.

The comment string contains several placeholders, including one called `%serverTranslationResultMsg%`. When the Servlet sends a page containing this comment string, the placeholders in the string are replaced with all the information describing the requested translation and the result.

As previously described, this information is delimited into numerous fields and sub-fields. Each group of fields is delimited by its own delimiter character, allowing a program to parse the string and extract the information it needs, including the user-specified string if present.

The information comes from several DDE components, including the FME Server itself. The FME Server's `fmeServerConfig.txt` file defines `SUCCESS_RESPONSE` and `FAILURE_RESPONSE` strings which are passed back through the QServer to the Servlet for every translation. The Servlet places the relevant `_RESPONSE` string into the FME Server response field within the QServer response field of the comment string in the DDE result web page.

If the `fmeServerConfig.txt` file is modified and a user information string is added to the end of the `SUCCESS_RESPONSE` and/or `FAILURE_RESPONSE` strings defined in the `SAFE_VIEWER_NONDISPLAY_RESULT` subsection, this user information will appear within the comment string of the DDE result web page.

The user information should be placed at the end of the `_RESPONSE` string(s) and separated from the previous field by a "|" pipe-character delimiter. The information itself can be any string that's meaningful to the program that parses it out and uses it. A `<name>=<value>` syntax can be used, or some other syntax if desired.

For example, if the following is appended to the `SUCCESS_RESPONSE` string in the `fmeServerConfig.txt` file:

```
 ...ResultLifetime=!FME_RESULT_LIFETIME!|userMsg=Hello my
friend
```

and then a zip-download translation is performed, the comment string at the top of the resulting `fetchZipSuccess.html` page would include:

```
...ResultLifetime=3600|userMsg=Hello my friend
```

As a second example, if a reference to one of the available pseudo-variables is included in the user information, like this:

```
 ...ResultLifetime=!FME_RESULT_LIFETIME!|userMsg=Hello my
!FME_NUM_FEATURES_OUTPUT! friends
```

the comment string would include something like this:

```
...ResultLifetime=3600|userMsg=Hello my 57 friends
```

Note that the user information will appear within the comment string, not at the end, since the Servlet adds more information after the FME Server response fields.

Asynchronous Result Return

This section describes asynchronous translation result returns by the Servlet. *Asynchronous* returns are those that the client does not wait for. Instead, e-mail notification is sent to the user once results become available.[2]

Translation Request Filtering

The determination to return result notification asynchronously by e-mail is made by the Servlet *request filter*.

2. Conversely, a *synchronous* return is one that the client waits for after making the request. Synchronous returns were described in *Synchronous Result Return* on page 215.

The request filter allows filtering of translation requests based on request complexity. Requests exceeding a predefined complexity level will be rejected or delayed for processing during a user-defined off-peak period.

If a request is delayed, the Servlet immediately sends back a web page to the client indicating that this is the case and that e-mail notification will be sent to the user once results are available.

Default Request Filter

A default filter is supplied with DDE. However, users can write their own custom filter as a Java class that implements the `COM.safe.viewerservlet.IRequestFilter` interface. This can replace the default filter to perform custom request filtering based on user-specific requirements. The Java request filter interface is described in *Request Filter Interface* on page 223.

The supplied default request filter computes request complexity as a function of the weights of the requested themes and the spatial area covered by the request.

The following default algorithm is used to determine request complexity:

```
    Cr = (Ar/At) * Σ(Wi; i=1,N)
```

where:

```
   Cr   = request complexity
   Ar   = request area
   At   = total source data area
   Wi   = weight of requested theme i
   N    = number of requested themes
```

Theme weights are specified for each theme in the `config.csv` file.

If an off-peak period is defined, complex requests will be delayed for processing until then. If no off-peak period is defined, complex requests will be rejected.

Requests for GIF image display will always be rejected if they are complex.

Properties

The default request filter behavior is controlled by several properties in the Servlet properties file.

The relevant properties are:

- **`useRequestFilter`**

  Filtering can be turned on or off by setting `useRequestFilter` to `true` or `false`.

  By default, request filtering is turned OFF. It must be turned on by modifying the relevant Servlet properties file and specifying:

  ```
  useRequestFilter=true
  ```

  then restarting DDE.

- **`requestFilterParam`**

  The default request filter takes the `requestFilterParam` property as a parameter string. This string consists of one or three values separated by pipe character delimiters. The format is:

  ```
  requestFilterParam=<complexityLimit>_<prohibitedComplexity>|
  <offPeakStartTime>|<offPeakStopTime>
  ```

  The first value represents a pair of measures of translation request complexity above which a request is deemed to be complex. A request's complexity is computed using the algorithm described above and is compared to the `<complexityLimit>` value. The request is defined to be *complex* if its computed complexity exceeds this limit value; otherwise it is *simple*. Complex request processing is typically deferred to an off-peak time period.

  The second complexity limit value is `<prohibitedComplexity>`. This limit value is optional. If present it must be separated from the `<complexityLimit>` value by an underscore character. Translation requests whose complexity exceeds `<prohibitedComplexity>` (if specified) are deemed to be not only complex but also *rejected*, and the default request filter always returns a `REJECT` status for such requests, and the translation is not performed.

  The remaining two values are also optional. They must both be present or both be absent. They specify the earliest and latest hour & minute of the day (24-hour clock) that represent the delayed processing period during which complex requests are to be processed.

  The delayed times must have the following format:

  ```
  <earliestHourNumber>:<earliestMinuteNumber>|
  <latestHourNumber>:<latestMinuteNumber>
  ```

  The 24-hour format is used. Single-digit hour values *should not* have a leading zero. Single-digit minute values *should* have a leading zero. Midnight is specified as 0:00. Noon is 12:00.

  Examples:

  ```
  9:15|17:00 period from 9:15 AM to 5:00 PM
  1:30|7:45 period from 1:30 AM to 7:45 AM
  0:00|12:00 period from midnight to noon
  ```

  If the delayed period values are absent, the filter assumes that no such period is available for processing complex requests and all such requests are rejected.

Custom Request Filters

Users can write their own custom request filters. Custom filters can replace the default filter to perform custom request filtering based on user-specific requirements.

When the Servlet is first started, it dynamically loads its request filter's class definition and creates a single instance of the filter. The Servlet must be told which request filter to use. This is accomplished by setting the value of the Servlet property `requestFilterClassName` to the classname of the desired request filter.

The filter's class definition must be accessible via the Java `CLASSPATH` environment variable.

Note: The Servlet uses multiple threads to process requests, but it dynamically loads, creates and uses a single instance of the request filter which is used by all threads. As a result, custom request filters must have the following characteristics:

- Must be written so as to be thread-safe. Any instance variables that a filter uses must be assigned values at creation/initialization time only, and subsequent references to these variables must be read-only.
- Must supply a constructor method that takes no parameters.

A custom request filter can also make use of a filter parameter string whose format and content can be set to whatever is desired by the filter. The value of this parameter string is set by a Servlet property called `requestFilterParam` located in the relevant Servlet properties file.

The default DDE request filter class is `COM.safe.viewerservlet.DefaultRequestFilter`. This class definition is located in the standard `spatialDirect.jar` file distributed with DDE.

The following sections describe the request filter interface.

Request Filter Interface

A custom request filter is written as a Java class that implements the `COM.safe.viewerservlet.IRequestFilter` interface. This interface is:

```
package COM.safe.viewerservlet;

import COM.safe.utility.*;

import java.util.*;

public interface IRequestFilter
{
  public int init(Hashtable filterInitParams);
  public String checkRequest(Hashtable requestParams);
}
```

The following sections describe the interface methods.

public int init(Hashtable filterInitParams)

The `init` method is invoked on the filter immediately after it is created by the Servlet. The filter must supply a parameterless constructor for its creation.

Input Arguments

The `init` method takes a Hashtable input argument containing the following information which can be used in its subsequent operation:

- The Servlet creating the filter. Obtained with:

  ```
  (ViewerServlet)filterInitParams.get("translationServlet");
  ```

- The Hashtable containing the theme weight values. Obtained with:

  ```
  (Hashtable)filterInitParams.get("themeWeights");
  ```

- The message logger used to write entries to the Servlet's logfile. Obtained with:

  ```
  (MessageLogger)filterInitParams.get("messageLogger");
  ```

- The request filter parameter string. The value of this is set in the servlet's properties file using the `requestFilterParam` property. Obtained with:

  ```
  (String)filterInitParams.get("requestFilterParam");
  ```

Return Argument

The `init` method must return one of the following integer value constants:

```
kVSRequestFilterInitFailed

kVSSuccess
```

These constants are available in the `COM.safe.viewerservlet.IViewerServletConstants` interface.

public String checkRequest(Hashtable requestParams)

The `checkRequest` method is invoked on the filter for every request received by the Servlet (assuming request filtering is enabled).

Input Arguments

The `checkRequest` method takes a Hashtable input argument containing the following information which can be used by the filter in its operation:

- The selected themes string containing the names of the requested themes separated by spaces. Obtained with:

  ```
  (String)requestParams.get("selectedThemes");
  ```

- The `COM.safe.utility.BoundsBox` containing the coordinates of the requested search rectangle. The coordinates represent the x and y coordinates of the rectangle's diagonally opposite corners. Obtained with:

  ```
  (BoundsBox)requestParams.get("searchRect");
  ```

The following methods return the `double` coordinate values in a `BoundsBox`: `getMinX() getMaxX() getMinY() getMaxY()`.

- The output format type string identifying the type of output format that the request is wanting to receive. Obtained with:

  ```
  (String)requestParams.get("outputFormatType");
  ```

  The returned string value will be one of the following:

  ```
  dataFile
  dataDir
  displayFile
  ```

- The output format tag string identifying the output format that the request is wanting to receive. Obtained with:

  ```
  (String)requestParams.get("outputFormatTag");
  ```

  The returned string value will be one which is defined in the following file:

  ```
  <DDEInstallDir>\safeViewerHTML\supportData\
  outputFormats.csv
  ```

  For example, `gif` is the tag identifying GIF format files, `shp` identifies ESRI Shape format files and `tab` identifies MapInfo format files.

- The `userInfo` Properties object containing user information. Obtained with:

  ```
  (Properties)requestParams.get("userInfo");
  ```

  The primary data in `userInfo` is the `name` property, whose value identifies the user's DDE session log-in ID.

- The `formParams` Hashtable containing the current values of either the request Order Form fields, or the remote fetch URL parameters. Obtained with:

  ```
  (Hashtable)requestParams.get("formParams");
  ```

- The Hashtable containing the theme weight values. The table key is the theme name and the value is the theme's weight as a `Double`. Obtained with:

  ```
  (Hashtable)requestParams.get("themeWeights");
  ```

- The `double` representing the minimum x coordinate value of the source data. Obtained with:

  ```
  ((Double)requestParams.get("minXSearchLimit")).doubleValue()
  ;
  ```

- The `double` representing the maximum x coordinate value of the source data. Obtained with:

  ```
  ((Double)requestParams.get("maxXSearchLimit")).doubleValue()
  ;
  ```

- The `double` representing the minimum y coordinate value of the source data. Obtained with:

  ```
  ((Double)requestParams.get("minYSearchLimit")).doubleValue()
  ;
  ```

- The `double` representing the maximum y coordinate value of the source data. Obtained with:

  ```
  ((Double)requestParams.get("maxYSearchLimit")).doubleValue()
  ;
  ```

- The StringBuffer containing the entire FME translation command string that is to be sent to the FME. Obtained with:

  ```
  (StringBuffer)requestParams.get("translationCommand");
  ```

Return Argument

The `checkRequest` method must return one of the String values listed below.

Each return consists of a keyword followed by a pipe-character delimiter, followed by the substring `Complexity=<complexityValue>`, where `<complexityValue>` represents the numerical value of the complexity as computed by the filter.

Here is an example return string: `DELAY_COMPLEX|Complexity=14.8`

The valid return strings are as follows:

- `NOW_SIMPLE|Complexity=<complexityValue>`

  The request is simple and should be processed now.

- `DELAY_SIMPLE|Complexity=<complexityValue>`

The request is simple and should be delayed for processing until a future time which is specified in the `formParams` Hashtable as the value of the `timeRequested` key, which may be the starting time of the predefined off-peak processing period.

- `NOW_COMPLEX|Complexity=<complexityValue>`

  The request is complex and should be processed now.

- `DELAY_COMPLEX|Complexity=<complexityValue>`

  The request is complex and should be delayed for processing until a future time which is specified in the `formParams` Hashtable as the value of the `timeRequested` key, which may be the starting time of the predefined off-peak processing period.

- `REJECT|Complexity=<complexityValue>`

  The request should be rejected for whatever reason is determined by the filter.

Custom Request Filter Class

A custom request filter class that implements the request filter interface must have a parameterless constructor for the Servlet to use when instantiating the filter.

A custom filter class would typically have the following declaration:

```
package COM.safe.viewerservlet;
import COM.safe.utility.*;
public class CustomRequestFilter implements
IRequestFilter,IViewerServletConstants,IUtilityConstants
{
custom filter code here...
}
```

Specifying Times for Delayed Processing

For `DELAY` return types, all request filters must ensure that `formParams` contains an entry whose key is `timeRequested` and whose value is a String representing the time to start processing the request. The format of this time string is defined using the Java `SimpleDateFormat` class with the following format specifier:

```
E-dd-MMM-yyyy hh:mm:ss a
```

This format specifier string is available as the following public constant value:

```
COM.safe.utility.SafeDateTimeFormatter.SAFE_DATETIME_FORMAT
```

An example of a `SAFE_DATETIME_FORMAT` time string is:

```
Wed-28-Jun-2000 09:26:03 AM
```

Result Notification via E-mail

If a request is determined to be complex (and/or possibly delayed for processing), a web page is immediately displayed indicating that the user will be notified by e-mail when the results are available. The e-mail address is obtained from the user during log-in, or from a remote fetch URL.

The value of the servlet property `defaultNotificationEmailAddress` can be set to a fallback default e-mail address to send notifications to if no other e-mail address is specified.

E-mail Message Content

The e-mail message itself contains URLs pointing to the translation results and the log file, and an indication of how long the results will be available. E-mail notifications of translation failures are also included, and give the reason for failure and a URL pointing to the translation log file if available.

Site-Specific Message Content

Optional site-specific content can be defined and included within the standard message body of the e-mail if desired. Both static and dynamic content can be defined as described below.

Static Content

Static content in the form of a header and/or a footer can be included in the e-mail message if desired. This static content is available for e-mail notifications resulting from interactive browser requests and from remote fetch URLs.

The DDE distribution includes the following two files for inserting static headers and footers:

```
<DDEInstallDir>/safeViewerHTML/supportData/
  notificationEmailHeader.txt

<DDEInstallDir>/safeViewerHTML/supportData/
  notificationEmailFooter.txt
```

Any text content in the above header file will be automatically inserted at the beginning of every notification e-mail message. Any text content in the above footer file will be inserted at the end of every notification e-mail message.

Both files are initially empty, and no header or footer text will appear in the message. Manually editing either or both of these files and entering the desired site-specific text will cause the Servlet to include that text as a header/footer as appropriate in the notification e-mail message.

Dynamic Content

Dynamic content can be inserted into e-mail notifications from two possible sources.

From Remote Fetch URLs

If the optional `description` parameter is specified in a remote fetch URL, the description value string will be inserted immediately before the standard message body (and after the static header if defined) in e-mail notification messages.

From Custom Message Body Files

The standard notification message body can be replaced by a custom message body. The same approach that's used for custom email header/footer content is followed for custom body content. To do custom body messaging, the following two text files must exist:

```
<DDEInstallDir>/safeViewerHTML/supportData/
  notificationEmailSuccessBody.txt

<DDEInstallDir>/safeViewerHTML/supportData/
  notificationEmailFailureBody.txt
```

If either or both of these two message body files are absent or empty, the default standard message body text is used instead (described below).

If present and non-empty, the content of these two files is used as the message body in the notification emails for translation success and failure respectively. (If defined, header and footer text is also added to this body).

The content can be anything desired. In addition to site-specific static content, users have the option to embed predefined, named placeholders within their text. Every placeholder occurrence will be replaced at message-send time by its appropriate current value.

To use these placeholders, they must be delimited within the text by bracketing them with single percent-signs. (If a literal percent-sign is desired, use a double percent-sign: "%%"). Any percent-sign-bracketed name not recognized as one of the predefined placeholders is reproduced as-is in the output text.

Examples of these two files are shown below and include the required names of all the available placeholders and what each represents. Note that the available placeholders for the success message body differ from those available for the failure message body.

Custom message body text can make use of all, some or none of these placeholders as required.

Note that DDE reads in the content of these two files only at start-up time. Changing the content requires restarting DDE to make the modifications take effect.

notificationEmailSuccessBody.txt - Available Placeholders

• Webserver results directory	`%resultDir%`
• Zipped results filename	`%zipResult%`
• FME translation log filename	`%fmeLog%`
• Output format tag	`%outputFormatTag%`
• Results creation date & time	`%currentDateTime%`
• Results deletion date & time	`%resultsDeletionTime%`
• Time-zone name (long)	`%timeZoneNameLong%`
• Time-zone name (short	`%timeZoneNameShort%`
• Name of user making translation reques	`%userName%`
• Description in remote fetch URL (if present)	`%description%`
• Specialized information message (if available)	`%infoMsg%`

notificationEmailFailureBody.txt - Available Placeholders

• Reason for failure	`%reason%`
• Webserver results directory (if available)	`%resultDir%`
• FME translation log filename (if available)	`%fmeLog%`
• Failure date & time	`%currentDateTime%`
• Name of user making translation request (if available)	`%userName%`
• Description in remote fetch URL (if present)	`%description%`

Layout of Standard Message Body

If a custom message body is not specified, DDE uses a standard message body by default. The layout of the standard message body is as follows:

For success notifications:

```
<optional site-specific static header>

<optional site-specific description>

Your DDE <result format tag> results are available at <result
file URL>

They were created on <result creation datetime>

The translation log is available at <translation logfile URL>

These results will remain available until <result deletion
datetime>

<optional site-specific static footer>
```

For failure notifications:

```
<optional site-specific static header>

<optional site-specific description>

Your DDE request could not be performed.

The reason was: <reason message>

This occurred on <failure datetime>
```

If the logfile is available, the following line appears:

```
The translation log was created at <translation logfile URL>

<optional site-specific static footer>
```

CHAPTER

15

Process Monitor

The Process Monitor is a Java program that starts other programs and automatically restarts them if they terminate for any reason, thereby providing a degree of fault tolerance for a system. Environments having one or more programs that need to be continuously available can have these programs started, monitored and restarted as necessary by the Process Monitor.

Theory of Operation

The Process Monitor can be used to start and monitor any program that can be started using a command line on the system on which the Process Monitor itself is running. A single Process Monitor can start, monitor, and restart one or more programs on the same system.

Although multiple Process Monitors can be simultaneously run on one system, it’s more usual to run a single Process Monitor on each system that requires it.

Typically, the Process Monitor is used to start the Translation Servlet, the QServer and one or more of the FME Translation Servers. For example, in a network with three systems called Processor1, Processor2, and Processor3, the administrator may want to run the QServer and one FME Translation Server on Processor1, two FME Translation Servers on Processor2, and the Translation Servlet on Processor3. A Process Monitor would be started on each of the three systems with the following setup:

System	Process Monitor Starts
Processor1	QServer
	FME Translation Server 1
Processor2	FME Translation Server 2
	FME Translation Server 3
Processor3	Translation Servlet

The Process Monitor determines which programs to run and monitor by reading command entries from its configuration parameter file. The format of this file and the syntax of its command entries are described below. Programs are started in the order in which they appear in the configuration file, so any QServer entry should appear before any FME Server entries, since the latter need to register with an already running QServer.

Use of the Process Monitor is optional. Neither the Translation Servlet, QServer or the FME Translation Server requires the Process Monitor for its operation. The Process Monitor simply adds an auto-restart capability, on an as-desired basis for critical programs that must be continuously available.

However, in normal DDE installations the Process Monitor is automatically used to start and monitor all components and this is the recommended approach.

Process Monitor Installation

The Process Monitor application is contained in the `spatialDirect.jar` file in the DDE installation directory. The Java `CLASSPATH` environment variable must include the pathname to this `jar` file. The following component is required:

- Sun Microsystems Java Runtime Environment (JRE) Version 1.4.1 or later

This component is distributed as part of DDE.

Process Monitor Configuration

Configuration parameters are used to set various operating characteristics of the Process Monitor. Values for these parameters are read in from a configuration file when the Process Monitor is started. The values assigned via the configuration file can be modified on a site-specific basis to suit the operating environment.

The configuration parameter file is an ASCII text file containing one parameter assignment on each line. Each assignment consists of a parameter name, followed by an equal (=) sign, followed by the parameter's value. For example, the line:

```
ECHO_LOG_TO_CONSOLE=true
```

assigns the `ECHO_LOG_TO_CONSOLE` parameter a value of `true`.

The Process Monitor is case-sensitive to parameter names. Blank lines and lines beginning with the number sign `(#)` are treated as comments, and are ignored.

Changes to any parameter value in this file take effect onlywhen the Process Monitor is restarted.

The following parameters are included in the configuration file and must be named exactly as shown:

Parameter	Description
`ADMIN_PASSWORD`	Password string required of clients when sending admin commands. **Required.**
`ADMIN_PORT`	TCP/IP port on which to listen for admin requests such as shutdown. **Required.**
`ECHO_LOG_TO_CONSOLE`	A Boolean flag indicating whether or not to display, or echo, all messages sent to the message log file to the default system output device, or console, as well. **Required**. A value of `true` causes messages to be echoed to the console. Any other value prevents echoing.
`ENABLE_SHUTDOWN_HOOK`	Boolean flag indicating whether or not to enable programmatic shutdown of DDE if the latter is interactively terminated using Ctrl-C. **Deprecated - superseded by stopSD command.** Programmatic shutdown is enabled only if this parameter is present and assigned a value of "true". The default is to enable Ctrl-C programmatic shutdown for Windows systems and to disable it for Unix systems.
`LOG_FILENAME`	The pathname of the message log file. **Required**.
`MAX_LOGFILE_AGE_SECONDS`	The maximum allowable age in seconds of previous versions of log files. Any log files older than this are deleted. Deletion of older log files only occurs when the current log file exceeds `MAX_LOGFILE_LINES` in size and is closed. **Optional.** Value < 0 previous log files NEVER deleted. Value = 0 ALL previous log files deleted. Value > 0 previous log files older than specified value deleted. If absent, this parameter is assigned a default value of 604800 seconds (7 days).
`MAX_LOGFILE_LINES`	The maximum number of lines written to the current logfile, after which it is closed, followed by possible deletion of older ones and creation of a new one to continue on with. **Optional.** Value <= 0 size limiting is NOT in effect. The same log file is used and can grow indefinitely. Value > 0 size-limiting IS in effect using the number of lines specified. If absent, this parameter is assigned a default value of 3000 lines.

Parameter	Description
`SHUTDOWN_COMMAND`	Defines a command string that will be run during DDE shutdown. **Required.** This command is typically used during shutdown to invoke a batch procedure which in turn shuts down the servlet engine running the Translation Servlet.
`TRUSTED_CLIENT_IPS`	Comma-separated list of IP addresses in numeric n.n.n.n format of clients trusted to send admin commands. **Required.**

Command Entry Parameters

Command entry parameters are those that define the actual commands that the Process Monitor should start as separate processes and then monitor. Each parameter represents one command and its process. There can be one or more command parameters present in the configuration file. The format for these command parameters is:

```
CMD<tag1>=<command1>[|log]
CMD<tag2>=<command2>[|log]
CMD<tagN>=<commandN>[|log]
```

where `<tag#>` is a user-specified, alphanumeric substring identifying the command.

Here is an example:

```
CMDStartFMEService1="C:\\Program Files\\FME\\fme.exe"
REGISTER_SOCKET JIM 7070
```

The tag identifying this example command is `StartFMEService1`. This tag can be any string. All tags used in the configuration file must be unique.

When the Process Monitor encounters this parameter, it runs the command string as specified and monitors the process for termination. If the process terminates the Process Monitor restarts the process by executing the same command string again. The Process Monitor continues this behaviour for the duration of its run.

Optional Suffix Flags

Additionally, a command string can end with either or both of the optional pipe-delimited suffix flags described below. Their order of appearance is not significant, but if either is present it must appear at the end of the command string, separated from it (and each other if both are present) by a pipe character.

`log` flag

If a command string includes the `log` suffix flag, process output generated by the command will be logged to the message log file. If a command string does not include this suffix, its process output is not logged to the log file. This mechanism can be used to enable/disable process output on a per command basis.

When process output is logged, each output line will be prefixed by the `CMD<tag#>` string. This helps to identify which process command generated each output line.

The example command above would have no logging of its process output. To enable process output logging for it, the command would be modified to:

```
CMDStartFMEService1="C:\\Program Files\\FME\\fme.exe"
REGISTER_SOCKET JIM 7070|log
```

The resulting logged output lines from this process would be prefixed by `CMDStartFMEService1`.

<noRestartReturnInteger> flag

This suffix flag consists of an integer number representing a status value returned to the Process Monitor by the monitored process when the latter terminates.

If a command string specifies a value for this suffix flag, then if the monitored process terminates and returns the specified status value to the Process Monitor, the latter will NOT restart the terminated process.

If this suffix flag value is absent, or it is present but the terminating process returns a status value different from the specified value, the Process Monitor WILL restart the terminated process.

Here is the same example above using both the `noRestart` flag (specifying a return status value of 27) and the `log` flag:

```
CMDStartFMEService1="C:\\Program Files\\FME\\fme.exe"
REGISTER_SOCKET JIM 7070|log|27
```

Running FME Servers

The QServer configuration file has a parameter that is set to indicate whether or not FME Servers are being run by Process Monitors. This is the recommended configuration.

Note If one FME Server is run by a Process Monitor, then *all* FME Servers must be run by Process Monitors. It is not possible to have combinations where some FME Servers are run by Process Monitors and others run on their own.

Process Monitor Start-up and Shutdown

The Process Monitor is the preferred method for automatic, sequenced start-up of all components of DDE, on all operating systems. The Process Monitor itself can be started manually or automatically.

Manual Start-up

Manual startup of the Process Monitor can be performed from the command line and takes one argument. This argument is the pathname of the configuration parameter file described above.

The general form of the Process Monitor start-up command is:

```
java COM.safe.processwatcher.FaultTolerator
<config-file-path>
```

Automatic Start-up

The Process Monitor can be made to start automatically as a service, independent of a user account, when the system on which it resides is rebooted. *Appendix B, DDE as a Windows Service*, describes how to set up the Process Monitor as a service on Windows.

Automatic start-up can be implemented on UNIX in a number of site-specific ways, such as adding the start-up command as an entry to the `/etc/init.d` file.

Shutdown

If manually started, it is recommended that the Process Monitor be shut down as part of the overall DDE shutdown procedure. Please see Chapter 2 for details on DDE shutdown.

If the Process Monitor was started automatically as a service, then the shutdown mechanism provided by that service should be used to shut down the Process Monitor.

Process Monitor Logging

During operation, the Process Monitor generates a log file to which it writes various information messages. This is an ASCII text file that can be examined with any text editor.

The log file pathname is specified in the configuration file. If log file size-limiting is in effect, the actual pathname of the current log file is this base pathname plus an integer suffix representing the number of milliseconds since the Java epoch. If size-limiting is not in effect, the log file pathname is as-specified in the configuration file.

Interfacing with the Process Monitor

There is currently no publicly exposed API to the Process Monitor and therefore no mechanism for external applications to interface with it.

CHAPTER 16

Servlet Engine Configuration

The default DDE configuration uses one of two servlet engines – Tomcat or ServletRunner – to run the Translation Servlet, both of which are bundled with the DDE distribution.

In this configuration, either ServletRunner or Tomcat is being used as an independent servlet engine, and its only task is to run the Translation Servlet. That is, the servlet engine is not tied to any particular web server (there is no explicit or implicit link to the Apache web server, for example) and it is started by the Process Monitor as an independent component of the DDE system.

This means that no matter which web server is being used on a DDE system, you may choose to use the default, bundled servlet engine for the Translation Servlet (recommended) or, if you wish to retain consistency with current servlet installations, you may replace ServletRunner/Tomcat with another servlet engine.

Background

When the servlet engine is started (ServletRunner or Tomcat is started by the Process Monitor in the default configuration), it listens on its default network port for requests. These requests specify the name of the servlet that the servlet engine is to run.

There are two methods of sending requests to the servlet engine: directly to the engine's port or indirectly by forwarding them through the webserver.

Request Forwarding

The indirect request forwarding method allows requests to reach DDE when the latter is operating behind a firewall and only the default webserver port is available to receive requests. This is the default method used by DDE.

Translation requests sent to DDE's servlet engine (Tomcat or ServletRunner) are routed to the engine indirectly through the system webserver's public request port (default 80).

This routing is performed by a CGI Perl script called `spatialDirect.pl` which is copied during installation into a user-specified subdirectory within the webserver's main CGI script directory.

At runtime a translation request is sent like any other public request to the webserver at (by default) port 80. The request includes a reference to the `spatialDirect.pl` Perl script, causing the latter to process the request. This processing consists of forwarding the request on to the DDE servlet engine which is listening on its own port (8194 by default). The latter then hands the request to the Translation Servlet as usual.

The same `spatialDirect.pl` Perl script waits for a response from the Translation Servlet and sends it back to the client that originated the request.

Please note that translation requests can still be sent directly to the servlet engine port rather than to the webserver public port for forwarding. However, the forwarding mechanism allows translation requests to be received by DDE when the latter is operating behind a firewall.

When a user logs in to the DDE system via a browser using the Translation Servlet's default log-in web page, the browser sends a URL request to the DDE CGI forwarding script which routes it to the servlet engine, specifying that the Translation Servlet is to be run. The servlet engine loads the Translation Servlet, initializes it, and starts it.

Subsequent communication takes place with the Translation Servlet through the servlet engine/CGI forwarding script transparently. Although the servlet engine and CGI script are present, communication can be treated as if it were occurring directly between the browser and the Translation Servlet.

When the servlet engine initializes the Translation Servlet, it reads a number of Translation Servlet configuration property values which are either defined in a file maintained for this purpose (e.g. `servlets.properties`) or as servlet engine configuration parameters, and makes them available to the Translation Servlet at runtime.

Translation Servlet behaviour is largely determined by its configuration property values and can be modified by changing these values and restarting the servlet engine (usually accomplished by restarting either DDE or the web server). After the servlet engine restarts, the next request sent to it that specifies the Translation Servlet will cause the latter to be reloaded and re-initialized with the new property values.

Java Version Incompatability Issue

If a servlet engine other than the Tomcat or ServletRunner engines bundled with DDE is used to run the Servlet, an issue of Java version incompatability may arise.

If the Java version used by the servlet engine (or servlet-enabled webserver) to run the Servlet is earlier than the Java version used to compile the Servlet, the engine may be unable to start or run the Servlet successfully.

At the time of writing, all Java components of DDE are compiled using Java V1.4.1. If the servlet engine uses Java V1.4.n, it should be able to run the Servlet without problems. However, if the engine uses Java V1.3.n or earlier, it will likely be unable to run the Servlet successfully.

This issue can arise with some enterprise-level webservers which use Java V1.3 to run servlets.

To avoid this situation, the DDE distribution includes a pair of `jar` files containing Servlet-related code compiled under Java V1.3 which should be runnable by servlet engines requiring Java V1.3-based servlets.

The jar files are located in:

```
<DDEInstallDir>\compatability\servletForJavaV1_3
```

and are called:

```
spatialDirectServletForJavaV1_3.jar
SafeCommerceServletForJavaV1_3.jar
```

These are Java V1.3-based counterparts to the standard V1.4-based `spatialDirect.jar` and `SafeCommerce.jar` files.

When configuring Java V1.3-based servlet engines (or servlet-enabled webservers) to run the DDE Servlet, the configuration parameters should specify these Java V1.3-based `jar` files rather than the standard ones.

ServletRunner Servlet Engine

The ServletRunner is a servlet engine implemented as a Java program supplied by Sun Microsystems as part of their Java Servlet Development Kit (JSDK). It is included as part of the default DDE installation for the following platforms:

- Solaris
- HP-UX

- AIX

When running DDE in its default configuration, you should not need to make any modifications to the ServletRunner installation. However, if you wish to do so, you will find the ServletRunner configuration file at: `<DDEInstallDir>/Jsdk/default.cfg`. This configuration file specifies, among other things, the network port on which ServletRunner listens for requests.

When the ServletRunner initializes the Translation Servlet, it reads a number of Translation Servlet configuration property values from a file maintained for this purpose (`<DDEInstallDir>/Jsdk/webpages/WEB-INF/servlets.properties`) and makes them available to the Translation Servlet at runtime. If modifications are made to this configuration file then ServletRunner will need to be restarted before they take effect.

For more information on the Translation Servlet configuration properties, refer to *Servlet Properties* on page 202.

Automatic Start-up

If ServletRunner is to be started automatically by the Process Monitor (this is the recommended procedure), the ServletRunner start-up command should be placed in the Process Monitor's configuration parameter file (this is done by default during the DDE installation process).

Manual Start-up

Manual Translation Servlet start-up (not recommended) is indirectly achieved by starting ServletRunner. This is done by running ServletRunner’s supplied `startserver` UNIX command file located in the `Jsdk` subdirectory located in DDE’s installation directory.

Automatic Shutdown

It is recommended that the Translation Servlet be shut down as part of the overall DDE shutdown procedure. Please see Chapter 2 for details on DDE shutdown.

Manual Shutdown

Manual Translation Servlet shutdown (not recommended) is accomplished by stopping ServletRunner. This is done by running the `stopserver` UNIX command file located in the `Jsdk` subdirectory of DDE’s root installation directory.

Tomcat Servlet Engine

Jakarta Tomcat is included as part of the default DDE installation for the following platforms:

- Windows
- Linux

When running DDE in its default configuration, you should not need to make any modifications to the Tomcat installation. However, if you wish to do so, you will find the Tomcat configuration files in: `<DDEInstallDir>/tomcat/conf/`. The configuration file (`server.xml`) in this directory specifies, among other things, the network port on which Tomcat listens for requests.

Configuration properties for the Translation Servlet are stored in the Tomcat `web.xml` file. This file is used to set the configuration properties for the DDE Translation Servlet. Property names and values are specified using XML tag syntax.

In the default DDE configuration, this properties file is located in:

```
<DDEInstallDir>/tomcat/webapps/servlet/WEB-INF/web.xml
```

For more information on the Translation Servlet configuration properties, refer to *Servlet Properties* on page 202.

DDE / Tomcat Port Changes

The following port changes are made in the Tomcat installation used by DDE. The Tomcat default ports are changed to the DDE default ports as shown. All modified Tomcat files are located in `<DDEInstallDir>/tomcat/conf.`

For Windows.

Tomcat File	Port Function	Tomcat Default Port	SD Default Port	SD Template Placeholder
`server.xml`	non-SSL Coyote HTTP/1.1 Connector	8080	8194	`<servletPort>`
	Coyote/JK2 AJP 1.3 Connector	8009	8195	`<ajp13Port>`

The Windows DDE installation dialog refers to the Tomcat default port 8009 as the *Coyote/JK2 AJP 1.3 Connector port*.

For Linux.

Tomcat File	Port Parameter Name	Port Function	Tomcat Default Port	SD Default Port	SD Template Placeholder
`server.xml`	`HttpConnectionHandler` port	servlet	8080	8194	`<servletPort>`
	`Ajp12ConnectionHandler port`	AJP12	8007	8195	`<ajp12Port>`
`workers.properties`	`worker.ajp12.port`	AJP12	8007	8195	`<ajp12Port>`
	`worker.ajp13.port`	AJP13	8009	8196	`<ajp13Port>`
`tomcat.properties`	`port`	JServ	8007	8195	`<ajp12Port>`
`wrapper.properties`	`wrapper.shutdown_port`	shutdown	8007	8195	`<ajp12Port>`
`tomcat.conf`	`ApJServDefaultPort`	JServ default	8007	8195	`<ajp12Port>`

The Linux DDE installation prompts refer to Tomcat default port 8007 as the *Apache Jserv AJP12 port,* and to the 8009 port as the *Apache Jserv AJP13 port.*

Manual Start-up

Manual Translation Servlet start-up (not recommended) is indirectly achieved when Tomcat is started. This is done by running Tomcat's supplied `startup.bat` Windows batch file or the `tomcat.sh` UNIX shell script and passing it the "run" parameter.

For Windows:

```
<DDEInstallDir>\tomcat\bin\startup.bat
```

For Linux:

```
<DDEInstallDir>/tomcat/bin/tomcat.sh run
```

Automatic Start-up

If Tomcat is to be started automatically by the Process Monitor (this is the recommended procedure), place the Tomcat start-up command in the Process Monitor's configuration parameter file (this is done by default during the DDE installation process).

Automatic Shutdown

It is recommended that the Translation Servlet be shut down as part of the overall DDE shutdown procedure. Please see Chapter 2 for details on DDE shutdown.

Manual Shutdown

Manual Translation Servlet shutdown (not recommended) is indirectly achieved by shutting down Tomcat. This is done by running Tomcat's supplied `shutdown.bat` Windows batch file or the `tomcat.sh` UNIX shell script and passing it the `stop` parameter.

For Windows:

```
<DDEInstallDir>\tomcat\bin\shutdown.bat
```

For Linux:

```
<DDEInstallDir>/tomcat/bin/tomcat.sh stop
```

Configuring Other Servlet Engines

The following sections describe how to configure various other servlet engines to run the Translation Servlet. Please note that although this is possible to do, it is not required, since the same functionality is provided by the servlet engines already bundled with DDE.

Moreover, the DDE installation automatically installs and configures the appropriate bundled engine to run the Translation Servlet transparently, avoiding the additional configuration necessary when setting up an external engine to run the Servlet.

The general recommendation is to use the automatically installed and configured servlet engines bundled with DDE, unless there is a pressing need to run the Servlet within an existing external engine environment.

iPlanet Web Server

iPlanet (also known as SunOne) is an integrated web server with the ability to run Java servlets internally. It is an extremely complex and powerful web solution with many different configurations possible. The intent of the following sections is to provide basic instruction on configuration of certain aspects of the iPlanet Web Server required specifically for the configuration of the DDE Translation Servlet.

Starting the Administration Server Management Console

Most configuration of the iPlanet Web Server is achieved through the Administration Server. Please follow the directions below to start the Administration Server.

1 Enter the following URL in your web browser:

```
http://<webservername>:<serveradminport>

e.g. http://yoda:8888
```

2 Type the requested username and password and click the OK button.

3 At the Administration Server page, select your server from the drop-down list and click the Manage button.

Configuring the Translation Servlet

1 Start the Administration Server management console as described above.

2 Select the Servlets tab at the top of the window.

3 Select the Configure Servlet Attributes button on the left of the window and fill in the fields as appropriate for your installation (you may have to replace existing field contents):

```
Servlet Name: sdTranslationServlet
Servlet Code (class name): COM.safe.viewerservlet.ViewerServlet
Servlet Classpath: <DDEInstallDir>/spatialDirect.jar
Servlet Args: …*
```

* The `Servlet Args` field should be populated with the name-value pairs defined in the servlet properties file that would normally be used by the default servlet engine bundled with DDE (Tomcat or ServletRunner). For more information on the Translation Servlet configuration properties, refer to *Servlet Properties* on page 202.

4 Click OK, then Save and Apply.

5 Click OK in the Success screen.

6 Select the Configure Servlet Virtual Path Translation button on the left of the window and fill in the fields as follows (you may have to replace existing field contents):

```
Virtual Path: /servlet/translationServlet
Servlet Name: sdTranslationServlet
```

Ensure that there is a slash (/) preceding the virtual path definition.

7 Click OK, then click Save and Apply.

8 Click OK in the Success screen.

Redefining the Servlet Port

If you have performed a standard installation of DDE, then all of your web pages will be attempting to locate the Translation Servlet on port 8194. By default, iPlanet web server uses port 80 for non-secure web sites and port 443 for secure web sites. You will need to change all references to port 8194 in your DDE web pages to make use of either port 80 or 443, as appropriate.

1 Change directory to: `<iPlanet document root>/ safeViewerHTML`

2 Find all files which contain a reference to port 8194 and replace the reference with 80 or 443 as appropriate.

Tip

On Windows systems `<iPlanet document root>` is normally found at: `C:\Netscape\Server4\docs.`

Standard Classpath Additions

You are required to inform iPlanet where the security and commerce `.jar` files can be found. To do this:

1 Start the Administration Server management console as described above.

2 Select the Servlets tab at the top of the window.

3 Select the Configure JVM Attributes button on the left of the window and add the following paths to the semicolon-separated list:

```
<DDEInstallDir>/security/jsse.jar
<DDEInstallDir>/security/jnet.jar
<DDEInstallDir>/security/jcert.jar
<DDEInstallDir>/commerce/SafeCommerce.jar
<DDEInstallDir>/commerce/Verisign.jar
```

4 Click OK, then click Save and Apply.

5 Click OK in the Success screen.

JavaMail E-mail Notification

If the request filtering option is set to `true,` then the JavaMail e-mail notification must be configured in iPlanet. To do this, follow the steps outlined below:

1 Start the Administration Server management console as described above.

2 Select the Servlets tab at the top of the window.

3 Select the Configure JVM Attributes button on the left of the window and add the following paths to the semicolon-separated list:

```
<DDEInstallDir>/util/activation.jar
<DDEInstallDir>/util/mail/mailapi.jar
<DDEInstallDir>/util/mail/smtp.jar
```

4 Click OK, then click Save and Apply.

5 Click OK in the Success screen.

ServletExec 3.0 Servlet Engine

Note: *<ServletExecInstallDir>* should be interpreted as references to your `ServletExec` installation directory, normally:
`C:/Program Files/New Atlanta/ServletExec ISAPI`
(For ServletExec 2.2 the application is installed under:
`C:/InetPub/ServletExec ISAPI`.)

Configuring the Translation Servlet

1 Start the ServletExec Admin utility: `Start | Program Files | New Atlanta | ServletExec 3.0 ISAPI | ServletExec Admin`

2 In the contents frame on the left of the window select Virtual Machine | classpath. (For ServletExec 2.2 select Advanced | VM Settings and then scroll down until you reach the section to “Enter additional directories to the Java VM classpath…”)

3 In the blank text field under the Java VM Classpath heading, enter the path to the Translation Servlet jar file:

```
<DDEInstallDir>/spatialDirect.jar
```

4 Click the Submit button.

5 In the contents frame on the left of the window, select Servlets | configure and then in the main window click the Add Servlet button.

6 Fill in the Servlet Name, Servlet Class and Initialization Parameters fields as:

```
Servlet Name: translationServlet
Servlet Class: COM.safe.viewerservlet.ViewerServlet
```

7 Click the Submit button.

Initialization parameters can be added in the Servlets configuration screen, however, due to the large number of parameters required for the `translationServlet` it is more simple to edit the configuration file for the servlet with a text editor and then restart the web server.

The configuration file, in a default installation, can be found at:

```
<ServletExecInstallDir>/ServletExec Data/default/
servlets.properties
```

8 Edit the configuration file's `# translationServlet servlet` section at the end of the file and add the line:

```
servlet.translationServlet.initArgs=initargs
```

`initargs` should be replaced with the name-value pairs defined in the servlet properties file that would normally be used by the default servlet engine bundled with DDE (Tomcat or ServletRunner). It must be a single comma-separated list of arguments in the format *`argumentName`*`="`*`argumentValue`*`"`

For details of the `ServletExec` configuration file format, refer to the ServletExec documentation.

For more information on the Translation Servlet configuration properties, refer to *Servlet Properties* on page 202.

Once you have made these changes, you will need to restart the web server.

Redefining the Servlet Port

If you have performed a standard installation of DDE, then all of your web pages will be attempting to locate the Translation Servlet on port 8194. By default, ServletExec (with IIS web server) uses port 80 for non-secure web sites and port 443 for secure web sites. You will need to change all references to port 8194 in your DDE web pages to make use of either port 80 or 443, as appropriate.

1 Change directory to: `<`*`web server document root`*`>/safeViewerHTML`

2 Find all files that contain a reference to port 8194 and replace the reference with 80 or 443 as appropriate.

Standard Classpath Additions

You are required to inform `ServletExec` where the security and commerce `.jar` files can be found. To do this:

1 Start the ServletExec Admin utility:

```
Start | Program Files | New Atlanta | ServletExec 3.0 ISAPI
| ServletExec Admin
```

2 In the contents frame on the left of the window select Virtual Machine | classpath. (For ServletExec 2.2 select Advanced | VM Settings and then scroll down until you reach the section to `"Enter additional directories to the Java VM classpath…")`

In the blank text field under the Java VM Classpath heading enter the paths to the commerce .jar files. After each path is added you will need to click the Submit button before adding another. The required paths are:

```
<DDEInstallDir>/security/jsse.jar

<DDEInstallDir>/security/jnet.jar

<DDEInstallDir>/security/jcert.jar

<DDEInstallDir>/commerce/SafeCommerce.jar

<DDEInstallDir>/commerce/Verisign.jar
```

3 Restart the web server.

JavaMail E-mail Notification

If the request filtering option is set to `true,` then the JavaMail e-mail notification must be configured in `ServletExec`. To do this, follow the steps below:

1 Start the ServletExec Admin utility:

```
Start | Program Files | New Atlanta | ServletExec 3.0 ISAPI |
ServletExec Admin
```

2 In the contents frame on the left of the window, select Virtual Machine | classpath. (For ServletExec 2.2, select Advanced | VM Settings and then scroll down until you reach the section to `"Enter additional directories to the Java VM classpath…").`

3 In the blank text field under the Java VM Classpath heading, enter the paths to the JavaMail jar files. After each path is added, you will need to click the Submit button before adding another. The required paths are:

```
<DDEInstallDir>/util/activation.jar

<DDEInstallDir>/util/mail/mailapi.jar

<DDEInstallDir>/util/mail/smtp.jar
```

4 Restart the web server.

JServ Servlet Engine

Notes:

- *`<JServRoot>`* should be interpreted as references to your `JServ` installation directory.
- *`<JServZoneFile>`* should be interpreted as references to your `JServ` Servlet zone configuration file: *`<JServRoot>`*`/servlets/zone.properties` (This is the default Servlet zone configuration. If you are using a custom servlet zone, such as the `esrimap` zone recommended by ESRI for the ArcIMS installation, then this should be interpreted as being used here.)

Configuring the Translation Servlet

1. Open the *<JServZoneFile>* for editing.
2. In the Repositories section, add the following entry:

   ```
   repositories=<DDEInstallDir>/spatialDirect.jar
   ```

3. In the Aliased Servlet Init Parameters section, add the following entry:

   ```
   servlet.Translation Servlet.initArgs=initArgs
   ```

   initargs should be replaced with the name-value pairs defined in the servlet properties file that would normally be used by the default servlet engine bundled with DDE (Tomcat or ServletRunner). It must be a single comma-separated list of arguments in the format *argumentName*="*argumentValue*". It may be split across multiple lines by using the line continuation character '\'.

For details of the Apache configuration file format, refer to the Apache documentation.

For more information on the Translation Servlet configuration properties, refer to *Servlet Properties* on page 202.

Redefining the Servlet Port

If you have performed a standard installation of DDE, then all of your web pages will be attempting to locate the Translation Servlet on port 8194. By default, `JServ` uses port 80 (the standard web server port). You will need to change all references to port 8194 in your DDE web pages to make use of port 80.

1. Change directory to: *<web server document root>*`/ safeViewerHTML`
2. Find all files that contain a reference to port 8194 and replace the reference with 80.

Standard Classpath Additions

You are required to inform `JServ` where the security and commerce `.jar` files can be found. To do this:

1. Open the *<JServZoneFile>* for editing.
2. In the Repositories section, add the following entries:

   ```
   repositories=<DDEInstallDir>/security/jsse.jar
   repositories=<DDEInstallDir>/security/jnet.jar
   repositories=<DDEInstallDir>/security/jcert.jar
   ```

```
repositories=<DDEInstallDir>/commerce/SafeCommerce.jar
repositories=<DDEInstallDir>/commerce/Verisign.jar
```

JavaMail E-mail Notification

If the request filtering option is set to `true`, then the JavaMail e-mail notification must be configured in `JServ`. To do this, follow the steps outlined below:

1. Open the *`<JServZoneFile>`* for editing.
2. In the Repositories section, add the following entries:

```
repositories=<DDEInstallDir>/util/activation.jar
repositories=<DDEInstallDir>/util/mail/mailapi.jar
repositories=<DDEInstallDir>/util/mail/smtp.jar
```

JRun Web Server 3.1 (Windows)

JRun is a comprehensive web application server and Java web server package. Although JRun can be configured to run your entire web site, this section only describes configuration of JRun for running the Translation Servlet. Please refer to your JRun documentation for information regarding other aspects of its configuration.

Starting the JRun Management Console

Most configuration of JRun is achieved through the Management Console. This can be started in two ways, described in point 1, below:

1. Start the Management Console.
   Enter the following URL in your web browser:

   ```
   http://<webservername>:<serveradminport>
   e.g. http://yoda:8000/
   ```

   Or

   Choose `Programs | JRun 3.1 | JRun Management Console` from the Windows Start menu.
2. Type the requested username and password and click the OK button.

Configuring the Translation Servlet

1. Start the Management Console as described above.
   Select your web server (or the default web server) by clicking its link on the left pane of the window. (By default, JRun installs the “JRun Default Server” which may be used for running the Translation Servlet.)

2 Select the Web Applications link in the left window pane, under your selected web server.

3 Select your configured web application, or use the Default User Application.

4 Select the Servlet Definitions link.

5 In the main window pane, click the Edit button to add a new servlet definition and enter values for the fields as defined below:

`Name:`	`translationServlet`
`Class Name:`	`COM.safe.viewerservlet.ViewerServlet`
`Display Name:`	`translationServlet`
`Init Arguments:`	`… *`

* The **Init Arguments** field should be populated with the name-value pairs defined in the servlet properties file that would normally be used by the default servlet engine bundled with DDE (Tomcat or ServletRunner).

For more information on the Translation Servlet configuration properties, refer to *Servlet Properties* on page 202.

WARNING: JRun cannot interpret arguments with a null value, therefore, you must ensure that all arguments supplied in the `Init Arguments` field contain a value. Take particular care with the `defaultNotificationEmailAddress` parameter which is likely to have a null value in default DDE installations.

6 Click the Update button.

7 Select the Virtual Mappings link in the left window pane.

8 In the main window pane, click the Edit button to add a new virtual mapping and enter values for the fields as defined below:

`Virtual Path:``/servlet`

`Mapping:``<`*`DDEInstallDir`*`>`

9 Click the Update button.

Redefining the Servlet Port

If you have performed a standard installation of DDE, then all of your web pages will be attempting to locate the Translation Servlet on port 8194. If you're using the JRun Default Server then DDE needs to be reconfigured to point to port 8100. (If you have defined your own JRun server, then you should substitute your own web server port number wherever you see 8100 in this section.)

Tip

To determine your web server's port number, click the web server's name in the left panel of the JRun Management Console and then click "JRun Web Server" under the Folder Options section of the main window pane. This should display a list of web server settings, including the Web Server Port.

To configure DDE to use server port 8100, you will need to change all references to port 8194 in your DDE web pages. To do this, follow the steps below:

1 Change directory to: `<`*`web server document root`*`>/ safeViewerHTML.`

2 Find all files that contain a reference to port 8194 and replace the reference with 8100.

3 Edit the Translation Servlet "Servlet Definitions" (described in *Configuring the Translation Servlet* on page 248). Update the Init Arguments parameter list so that the `servletURL` parameter refers to port 8100, not 8194.

4 Click the Update button.

Standard Classpath Additions

You are required to inform JRun where the security and commerce `.jar` files can be found. To do this:

1 Start the Management Console as described above.

2 Select your web server (or the default web server) by clicking its link on the left pane of the window.

3 Select the "Java Settings" link in the left window pane, under your selected web server.

4 Select the Classpath link in the main window pane.

5 Add the following entries to the classpath Input Field window:

```
<DDEInstallDir>\spatialDirect.jar
<DDEInstallDir>\security
<DDEInstallDir>\commerce
```

6 Click the Update button.

CHAPTER 17

Translation Request Processing

The FME Server component is responsible for performing the actual translation of source data to the specified output format.

In the default DDE configuration, FME Server translations are controlled by a predefined set of FME mapping files and other related script files. These mapping files include format-specific importers and exporters, as well as common utility INCLUDE files and Tcl scripts.

This chapter describes the processing of DDE translation requests as controlled by the default collection of mapping files shipped with DDE. Note that these files can be modified to provide customized translation behaviour that is specific to a user's environment.

Mapping Files

The FME Server is built upon FME core technology and, as such, fully supports the use of FME mapping files for performing translations and feature manipulation.

An FME mapping file can control complex geo-processing and translation tasks in any of the formats supported by FME. These tasks can include updating the source dataset, extracting a subset from the dataset and translating it into the requested format, or performing a spatial operation against a dataset, such as polygon formation or rubber sheeting.

The default mapping files bundled with DDE are set up to extract a subset of spatial data from a source dataset, and translate (and possibly reproject) this data to a desired output format. The other components of DDE manage the delivery of the translated results to the end user.

These mapping files make extensive use of macros whose values are assigned in the FME command string that is passed to the FME Server by client applications such as the Translation Servlet. When the latter is used, many of the macro values used by the mapping files originate in either the default Order form web page or the remote fetch URL sent to the Servlet. The Servlet inserts these macro values into the FME command string it builds, then passes it to the FME Server via the QServer.

Mapping File Locations

DDE's default set of mapping files is located in the `<DDEInstallDir>/translationControl` directory tree. Figure 17-1 shows the layout of this tree.

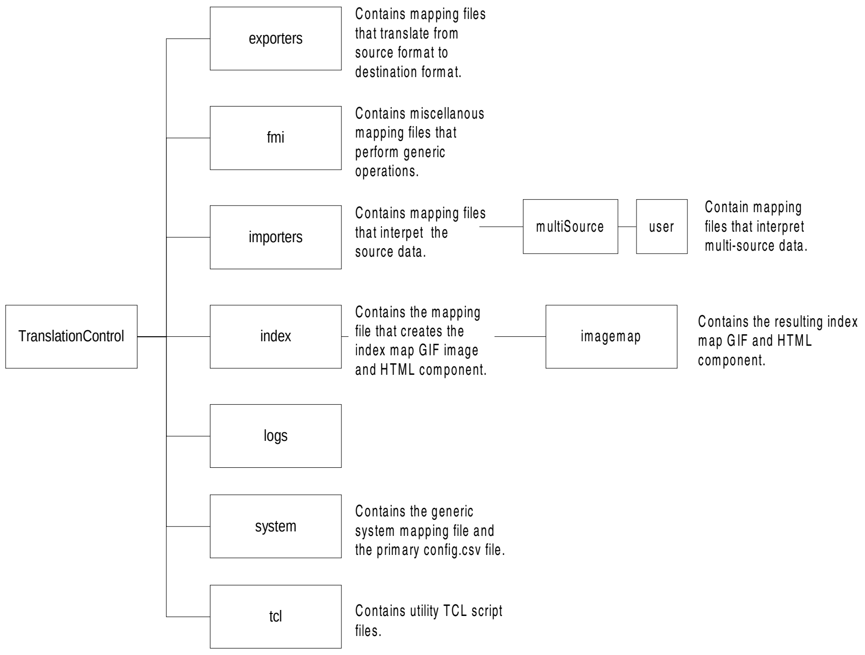

TranslationControl
exporters
Contains mapping files that translate from source format to destination format.
fmi
Contains miscellanous mapping files that perform generic operations.
importers
Contains mapping files that interpet the source data.
multiSource
user
Contain mapping files that interpret multi-source data.
index
Contains the mapping file that creates the index map GIF image and HTML component.
imagemap
Contains the resulting index map GIF and HTML component.
logs
system
Contains the generic system mapping file and the primary config.csv file.
tcl
Contains utility TCL script files.

FIGURE 17-1 **Translation Control Directory Layout**

Mapping File Control Flow

Figure 17-2 illustrates the general flow of control that occurs within the FME Server mapping files during a translation when using a single source of data. The numbers indicate the order of processing.

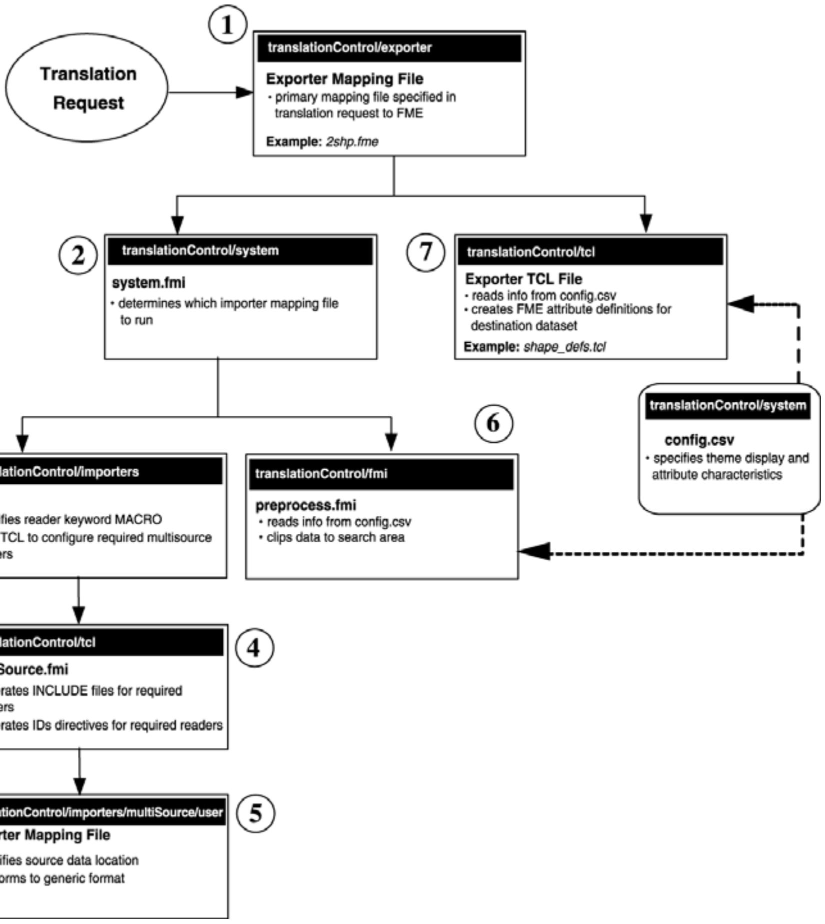

Translation Request
1
translationControl/exporter
Exporter Mapping File
• primary mapping file specified in translation request to FME
Example: 2shp.fme
2
translationControl/system
system.fmi
• determines which importer mapping file to run
7
translationControl/tcl
Exporter TCL File
• reads info from config.csv
• creates FME attribute definitions for destination dataset
Example: shape_defs.tcl
translationControl/system
config.csv
• specifies theme display and attribute characteristics
3
translationControl/importers
• specifies reader keyword MACRO
• calls TCL to configure required multisource readers
6
translationControl/fmi
preprocess.fmi
• reads info from config.csv
• clips data to search area
4
translationControl/tcl
multiSource.fmi
• generates INCLUDE files for required readers
• generates IDs directives for required readers
5
translationControl/importers/multiSource/user
Importer Mapping File
• specifies source data location
• tranforms to generic format

FIGURE 17-2 **Mapping File Control Flow**

Importer Mapping Files

To support source data that originates from multiple sources, a set of multi-source importer mapping files is used.

Note: All source data is defined in DDE using the multi-source mechanism. This is true even when only a single data source is being used. In this case it simply means that the source environment is defined as a multi-source configuration consisting of a single data source.

The multi-source importer mapping files are all located within `<DDEInstallDir>\translationControl` and are described as follows:

importers\multiSource.fmi

This file acts as the main importer mapping file when multi-source data is being used. It defines a macro that lists each user-defined source data reader keyword and its associated format type. It then runs the `tcl\configMultiSource.tcl` script.

tcl\configMultiSource.tcl

This Tcl script file does the following:

- dynamically generates INCLUDE statements for the required `<readerKeyword>.fmi` files defined by the user and located in `importers\multiSource\user`.
- dynamically generates INCLUDE statements for the required `<format>common.fmi` files located in `importers\multiSource`.
- dynamically generates `<readerKeyword>_IDs` directives for the required source datasets.

importers\multiSource

This directory contains the following files for each supported multi-source data format:

- `<format>config.fmi` INCLUDEd by the user-defined `<readerKeyword>.fmi` files.
- `<format>common.fmi` INCLUDEd by the `tcl\configMultiSource.tcl` script.

importers\multiSource\user

This directory contains the following files:

- `<format>template.fmi` Used to generate user-defined `<readerKeyword>.fmi` files. There is one template file for each supported multi-source data format.
- `<readerKeyword>.fmi` User-defined for each source dataset using the template files.

For a list of defined importer mapping file templates, see the table *Importer Mapping Files on page 122.*

Exporter Mapping Files

The default set of predefined exporter mapping files shipped with DDE allows writing to the following destination formats.

TABLE 17-1 Exporter Mapping Files

Mapping File	Supported Format
`2arcgen.fme`	ARC/INFO Generate
`2coverage.fme`	ARC/INFO Coverage
`2dxf12.fme`	AutoCAD DXF Rel 12
`2dxf14.fme`	AutoCAD DXF Rel 14
`2dxf2000.fme`	AutoCAD DXF 2000
`2dwg12.fme`	AutoCAD DWG Rel 12
`2dwg14.fme`	AutoCAD DWG Rel 14
`2dwg2000.fme`	AutoCAD DWG 2000
`2e00.fme`	ARC/INFO E00 Archive files
`2eps.fme`	Encapsulated PostScript (EPS)
`2esriGML.fme`	ESRI GML
`2gif.fme`	GIF raster images
`2gml2.fme`	GML 2 (Safe Schema)
`2grd.fme`	Penmetrics GRD
`2ieps.fme`	Illustrator Encapsulated PostScript (EPS)
`2igdsV7.fme`	MicroStation V7 Design Files (DGN)
`2igdsV8.fme`	MicroStation V8 Design Files (DGN) (Windows only)
`2mapinfo.fme`	MapInfo TAB (Native)
`2mif.fme`	MapInfo MID/MIF
`2png.fme`	PNG raster images
`2sdl.fme`	Autodesk MapGuide SDL
`2shp.fme`	ESRI Shape files
`2svg.fme`	SVG Scalable Vector Graphics (not available on AIX)
`2vml.fme`	VML
`2vrml.fme`	VRML
`2whitestar.fme`	Whitestar GES Cartographic

These exporter files are located in `<DDEInstallDir>\translationControl\exporters`

In addition to these supplied exporter mapping files, users can add their own customized mapping files to the DDE system to perform specialized translation output processing. Users can also modify the supplied mapping files if desired.

Translation Command String

The FME Server performs a translation according to the translation command string sent to it. In the full DDE default configuration, the Translation Servlet constructs this command string either from information in the default Order Form web page or from parameters within a remote fetch URL (described in *Remote Fetch URL Interface* on page 177). The resulting command is then sent to the FME Server via the QServer.

The command string includes FME macro definitions that are used within the various default mapping files. The structure of a translation command string as constructed within the default DDE environment is as follows:

```
<mappingfile> --THEMES  "<ThemeList>"               \
              --MINX          <Ordinate>            \
              --MINY          <Ordinate>            \
              --MAXX          <Ordinate>            \
              --MAXY          <Ordinate>            \
              --DestCoordSys  <CoordinateSystem>    \
              --PIXELS        <NumberofPixels>      \
              --DestDataSet   <ResultFile>          \
              --LogFile       <LogFileName>         \
              --TimeStamp     <Time>
```

As an example, the following command string causes the FME Server to run the `2shp.fme` mapping file, which extracts data from the source themes `ROADS` and `TAXAREAS` in the defined area, translates the data to Shape format, reprojects the data to UTM Zone 10, places the result in a file called `C:\temp\result` and sends log messages to `shp.log`:

```
2shp.fme      --THEMES"ROADS TAXAREAS"              \
              --MINX          -120                  \
              --MINY          49                    \
              --MAXX          -121                  \
              --MAXY          50                    \
              --DestCoordSys  UTM-10                \
              --DestDataSet   C:\temp\result        \
              --LogFile       shp.log               \
              --TimeStamp 973115943339
```

The `TimeStamp` macro value represents the number of milliseconds since the Java epoch. In DDE, the Translation Servlet calculates this value which is then used by the FME Server to generate unique file names for the translation results where necessary.

The macros used in the default translation command are listed in Table 17-2 .

TABLE 17-2 Default Translation Command Macros

Macro Name	Assigned By	Description
`DestCoordsys`	Servlet	The destination coordinate system into which the data will be transformed.
`DestDataset`	FME Server (specified in Server's config file)	The name of the destination dataset that contains the translation results.
`LogFile`	FME Server (specified in Server's config file)	The name of the log file that will be created for the translation.
`MAXX`	Servlet	The x coordinate of the rightmost bound of the clipping rectangle of the request.
`MAXY`	Servlet	The y coordinate of the highest bound of the clipping rectangle of the request.
`MINX`	Servlet	The x coordinate of the leftmost bound of the clipping rectangle of the request.
`MINY`	Servlet	The y coordinate of the lowest bound of the clipping rectangle of the request.
`PIXELS`	Servlet	The height and width in pixels of the GIF output image. This is only used when the result is specified to be a GIF.
`THEMES`	Servlet	A space delimited list of themes to process in the request, and must be enclosed in double quotes; for example, "A B".
`TimeStamp`	Servlet	The time the request was made and is used in as a component of the result filename where necessary. It is the number of milliseconds since the Java epoch.

GIF Image Output Characteristics

DDE provides default GIF image output through the FME's GIF Writer, whose behaviour is controlled by DDE's standard `2gif.fme` mapping file.

The following sections describe the characteristics of GIF output when this mapping file is used.

Generated Output Files

For each translation to GIF, the `2gif.fme` mapping file causes the GIF Writer to generate the actual GIF image file and two HTML files as follows:

Output File	Content
`<temp-filename>.gif`	actual GIF image
`<temp-filename>.gif.html`	HTML image map code
`<temp-filename>.gif.legend.html`	HTML legend code

Before the Translation Servlet downloads the default GIF display web page `(fetchGifSuccess.html)` to a browser, it inserts the contents of both HTML files into the page. The Servlet does this by connecting directly to the webserver and requesting the latter to send to it the HTML contents of both files for inclusion in the web page.

The page also includes a URL reference to the GIF image file itself, causing the image to be displayed within the page. The two HTML files are described in the sections below.

GIF Image Display Characteristics

Image Map

The `<temp-filename>.gif.html` file contains the HTML code that specifies the image map associated with the GIF image. This image map defines polygonal areas for each feature in the image and associates a JavaScript alert function call for each area. The argument for each function call is composed of the names and values of the attributes for the feature. The mapping file obtains this GIF attribute information from the `config.csv` file.

The creation of image map polygonal areas around non-polygonal features represents a form of buffering. The extent and fineness of this buffering as measured in pixels are controlled by the `GIF_IMAGE_MAP_BUFFER_SIZE`, `GIF_IMAGE_MAP_MIN_AREA` and `GIF_IMAGE_MAP_MIN_LINE_LENGTH` GIF Writer directives in the mapping file.

When the user clicks on a feature in the GIF image, the underlying image map polygon triggers the alert function, causing the browser to display a panel showing attribute name-value pairs associated with the feature.

Image Map Theme Restriction

By default, all themes whose `Gif Alt Label` value in `config.csv` is not a hyphen (-) will be represented in the image map. Clicking on the features of such themes will cause their attributes to be displayed as described above. All themes whose `Gif Alt Label` value is “-” will not be represented in the image map.

This default behaviour can be dynamically modified to allow only a single theme’s feature attributes to be displayed when clicked on, regardless of how many other themes are being displayed in the image.

This restriction is accomplished by assigning the desired theme name as the value for the `IMAGEMAP_THEME` FME macro. When this is done, only the single theme named by `IMAGEMAP_THEME` (whose `Gif Alt Label` value must not be “-”) will be enabled for clickable attribute display in the GIF image. Clicking on features for any other theme will be ignored.

The `IMAGEMAP_THEME` macro is typically specified as part of a *remote fetch URL*.

Legend

The `<temp-filename>.gif.legend.html` file contains the HTML code that defines the legend associated with the GIF image. The mapping file uses a Tcl script to generate this code. Each theme that appears in the GIF image has an associated entry in the legend. Each entry consists of a key colour box and the name of the theme.

Sizes

The size of point features in the GIF image is set using the `dotSize` macro. The mapping file sets this to a default value of 3 pixels.

The width and height of the GIF image are set to the same value in pixels, resulting in a square image display. The size value is obtained either from the interactive order form web page or from a parameter in a remote fetch URL.

GIF image pixels are set to be square, and those that are not in the ground range of the feature data are trimmed off.

Background Color

The GIF image background color is set to white.

Label Text

The following two macros are available to set GIF label text appearance:

`GifLabelColor` – all GIF label text is set to the color specified by this macro. The default color value is Black.

`GifMaxNumLabels` – this macro value sets the maximum number of labels allowed for a theme as a means of reducing image clutter. If a theme has more than this number of labels, then *no* labels are displayed for the theme. The default value is 150.

Drawing Order

The GIF Writer draws all polygons features first, followed by line, point and finally text features. The drawing order within each of these groups is determined by each theme's `GIF Precedence` value as specified in the `config.csv` file.

Within each geometric group, higher values of `GIF Precedence` cause a feature to be drawn above those having lower `GIF Precedence` values.

Visible Bounding Box

In addition to the actual features returned from a translation, the GIF image also displays the bounding box containing the features. This box is drawn using a grey dashed line.

Appendices

APPENDIX

A

QServer Administration

The QServerAPI allows third-party programs to control the QServer and to receive real-time statistical information from the QServer describing its usage and performance. It is the same API used for sending translation requests to the QServer and is described in *Interfacing with the QServer* on page 156.

Administration Commands

`QServerAdminTester`, and user-written programs in general, can send the following administration command strings to the QServer via the QServerAPI mechanism:

- `pause requests`
- `resume requests`
- `pause results`
- `resume results`
- `get request queue`
- `get result queue`
- `get requests`
- `get results`
- `get stats`

These administration command strings are analogous to the translation request command strings that are sent to the QServer using the same QServerAPI when making translation requests.

QServer AdminTester Program

DDE provides a program called `QServerAdminTester`, a utility that illustrates the use of the QServerAPI class to send *administration* commands to the DDE QServer (as distinct from translation requests).

The `QServerAdminTester` program is contained in the `spatialDirect.jar` file in the DDE installation directory.

Running the Program

`QServerAdminTester` is run from a command-line console and waits for the user to enter one of the above QServer administration commands. It sends this command to the QServer, waits for the QServer to return the result Transaction and displays the main result string to the console. It then waits again for the next command.

To start `QServerAdminTester`, enter the following on the command line:

```
java -classpath "<DDEInstallDir> spatialDirect.jar"
COM.safe.serviceprovider.QServerAdminTester <qServerHost>
<adminPort>
```

where <`qServerHost`> is the hostname of the system running the QServer and <`adminPort`> is the port on which the QServer is listening for administration commands (7072 by default).

`QServerAdminTester` is stopped by entering the `quit` command.

QServerTester

DDE provides a program called `QServerTester` that can be used to test a QServer installation. `QServerAPI` is the name of the API through which client applications communicate with the QServer. The `QServerTester` is contained in the `spatialDirect.jar` file in the DDE installation directory.

The `QServerTester` simulates one or more client applications, each making one or more transaction requests for FME Server translations through the QServer. The `QServerTester` in effect takes the place of the Translation Servlet as a client of the QServer.

QServerAPI Sample Files

The files in the DDE `config\QServer` directory include a sample FME mapping file called `qServerTest.fme` and a companion sample input data set called `test.dgn`. The latter is a small MicroStation Design file containing a few polygons, lines, circles, and text features.

The mapping file expects `test.dgn` to be in the same directory it's in, although this can be changed by modifying the mapping file, if desired. The mapping file translates the design file to ESRI Shape format, and outputs the resulting files to the same directory.

The `QServerTester` should be run using these files. The `<request-command>` parameter is the pathname of the `qServerTest.fme` file, as shown in the example below.

QServerTester Start-up

Once the QServer has been started, and one or more FME Servers have been started and have registered with it, the `QServerTester` can be run to send requests through the QServer to the waiting FME services.

The general form of a `QServerTester` startup command is:

```
java COM.safe.serviceprovider.QServerTester
<QServer-hostname> <QServer-request-port>
<num-simulated-clients> <num-requests-from-each-client>
<request-keyword> <request-command>
["<year> <month> <date> <hour> <minute>"]
```

Submission Time Syntax

The last parameter is optional, as indicated by its surrounding square brackets. This can be used to specify a time after which the QServer will submit the request for execution. The request is held until this time.

Each of the five required time fields is an integer representing the number of the year, month, date, and hour or minute of the desired time. The entire set of time values must be enclosed in double quotation marks. If this parameter is absent, or if it specifies a time in the past, transactions are submitted immediately.

The submission time syntax is as shown here:

- `<year>` = integer in the form yyyy
  examples: 1999 2000
- `<month>` = integer from 1-12
  examples: 1 (January) 5 (May)
- `<date>` = integer from 1-31
- `<hour>` = integer from 0-23 (24-hour clock)
- `<minute>` = integer from 1-59

Note that the submission time syntax described here is used only by the `QServerTester` program. In general, submission times are programmatically set via the QServerAPI and are specified as Date objects.

Example

Here is an example `QServerTester` start-up command:

```
java COM.safe.serviceprovider.QServerTester JIM 7071 10 5
"" C:\TEMP\qServerTest.fme "2002 6 9 15 15"
```

In this example, the hostname is `JIM` and the request port number is `7071`. This is the port on which the QServer running on `JIM` listens for client requests. This port value must match the value assigned to the `REQUEST_PORT` parameter in the QServer's configuration file.

This example sets the number of clients to 10, with each one making 5 translation requests. The request keyword is the null string and the request command (the actual translation command that the FME Server is to perform) is the pathname of the FME mapping file provided with the `QServerTester`. All clients make the same translation request each time. In this example the requests will be held for submission until 09-June-2002 at 3:15 PM.

Using the QServerTester

The following procedure can be performed when installing and testing the QServer application using the `QServerTester`. Performing this procedure affords a good first-test check on the installation. The following assumes installation on an Windows system, but the same approach is used on UNIX platforms as well. The test directory on all systems is assumed to be `C:\Temp`.

1. Copy all provided files into the test directory on the QServer system.
2. Copy the `qServerTest.fme` and `test.dgn` files into the test directory of each system that will provide an FME service.
3. Check the QServer's configuration file and adjust the values of any parameters if desired. In most cases the existing values can be used.
4. Open a DOS command window on the QServer system and start the QServer.
5. Open a DOS window on each system that provides an FME service and start the FME Server to register with the QServer.
6. Open another DOS command window on the QServer system and start the `QServerTester` with the desired parameters.

The `QServerTester` begins issuing requests using the specified number of clients and number of requests/client.

Each time an FME service receives a request, it logs it to its DOS window and performs the translation.

Each time a client receives a result, it logs it to the `QServerTester` DOS window. The logged information includes the result ID, result string, and timestamps.

Once all results have been returned, the `QServerTester` program ends. The QServer and the FME services remain running, waiting to process further requests.

The QServer log file can be examined to check on the operation.

Usage Notes

When using the `QServerTester`, the following points should be noted:

- The maximum number of clients is determined by per-process system resource limitations. On an NT V4 system this is typically on the order of 20.

  Attempts to create more than this approximate number of clients will fail when the QServerTester is used.
- If multiple requests are pending and there is more than one FME Server running *on the same system*, only one of the requests will be processed and the other services will abort. This is because every QServerTester client makes the same request. If multiple FME Servers are running on the same system, each will be given the same request, causing each one to attempt to read from and write to the same files at the same time. File access conflicts result and in most cases all but one of the services will fail.

  If multiple FME Servers and multiple pending requests are to be tested with the `QServerTester`, each FME service should be run on a separate system. This is a restriction of the `QServerTester`, and not of the QServer itself.

APPENDIX

B

DDE as a Windows Service

As an alternative to manual start-up, you can start DDE and run it as a Windows Service. This allows DDE to run independent of any user log-in session. This appendix outlines the procedures required to do this.

DDE Windows Service Start-up

The Process Monitor component of DDE is what is actually run as a Windows Service. During DDE installation on Windows, the Process Monitor is automatically installed (but not started) as a Windows Service and set to use the `<DDEInstallDir>\processMonitorConfig.txt` file.

To start the Process Monitor as a Windows Service, perform the following steps:

Note: Steps 1, 2 and 3 below are performed automatically when DDE is installed using the supplied InstallShield program. It is only necessary to manually perform these first three steps if the InstallShield program was not used to install DDE.

1 The `CLASSPATH` environment variable must be set at the system level. On NT this is done through the Windows control panel by clicking `Start | Settings | Control Panel | System | Environment.`

The system-level `CLASSPATH` value must be set to include the path name of the DDE installation's `spatialDirect.jar` file. The entire value assigned to `CLASSPATH` should not exceed 430 characters in length.

2 The `Path` environment variable must also be set at the system level using the same control panel sequence.

The system-level `Path` value must be set to include the path name of the DDE installation's `jre\bin\server` directory. This directory contains the `jvm.dll` Java Virtual machine file.

3 If any of these system environment settings are changed, the system must be rebooted to ensure that the changes take effect.

4 Open the Services Control Panel.
   - On NT, open the Control Panel, then click `Services`
   - On Windows 2000 and XP, open the Control Panel, then click `Administrative Tools | Services`

5 Scroll down the displayed list of services and select the `ArcIMSDDE` service.

6 On the Services Control Panel, click the Start button (NT) or right click the service name and select Start (Windows 2000/XP). The service should commence starting up after a few seconds. It proceeds to start the Process Monitor, which in turn starts the other DDE components installed on the system (QServer, FME Server, and Tomcat).

You should wait for approximately one more minute before using DDE to give the various component processes time to complete initialization.

To confirm that the DDE components have started, start the Task Manager and select the Processes tab. The display should include the following new processes, depending on which DDE components have been installed:

Image Name	Associated Program
`fme.exe`	`FME Server`
`java.exe`	`QServer`
`java.exe`	`Tomcat (servlet engine for Translation Servlet)`
`DDEService`	`DDE (Process Monitor) service`

When DDE's Process Monitor is started as a Windows Service in this way, it will remain running between log-in sessions, and requires restarting only if the system is rebooted. If the Service's `Startup Type` was set to `Automatic`, DDE will be automatically started during system boot-up without user intervention.

The figure below shows the default DDE Windows Service start-up sequence..

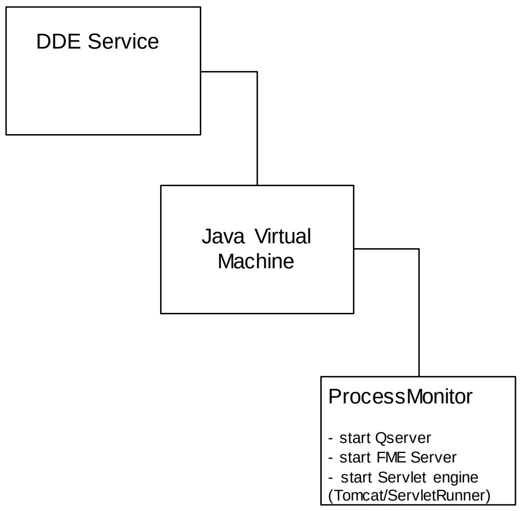


FIGURE B-1 **Default DDE Service Start-up Sequence**

DDE Windows Service Shutdown

When DDE is running as a service, it can be stopped by performing the following steps:

1. Open the Services Control Panel:
   - On NT, open the Control Panel, then click `Services`
   - On Windows 2000 and XP, open the Control Panel, then click `Administrative Tools | Services`
2. Scroll down the displayed list of services and select the `ArcIMSDDE` service, then click the Stop button (NT) or right click the service name and select Stop (Windows 2000/XP).
3. On Windows NT answer **Yes** to the confirmation panel.

All DDE processes will be automatically shut down within approximately one minute.

Uninstalling the DDE Windows Service

Note: You must uninstall the DDE Windows Service before uninstalling the DDE itself, since the service uninstallation procedure requires a small program that is contained within the DDE installation.

Running the DDE uninstallation procedure does not uninstall the DDE Windows Service. The Service must be uninstalled manually as follows.

Enter the following command in a DOS command window with Administrator privileges:

```
<DDEInstallDir>\WindowsService\DDEService -r
```

Manually Reinstalling the DDE Windows Service

It may sometimes be necessary to manually reinstall the DDE Service. For example, if DDE is being reinstalled on a system that already has an older version of the DDE Windows Service defined (as a result of a previous installation).

In these cases the reinstallation of DDE itself will *not* reinstall the DDE Windows Service. Instead, the existing DDE Service is kept and used for the new DDE installation. This means that the Process Monitor will continue to use the `processMonitorConfig.txt` file that was specified when the Service was first installed.

This will produce an error if the file is subsequently relocated.

To avoid this situation, the DDE Service should be reinstalled manually as follows:

1. Shut down the DDE Service if it is running.
2. Uninstall the DDE Service as described above.
3. Reinstall the DDE Service by entering the following command in a DOS command window with Administrator privileges, specifying the path of the `processMonitorConfig.txt` file to use (the default path is shown here):

   ```
   <DDEInstallDir>\WindowsService\DDEService -i <DDEInstall-
   Dir>\processMonitorConfig.txt
   ```

4. Restart the DDE Service.

APPENDIX C

TCP/IP Port Usage

The default DDE configuration uses the TCP/IP ports listed on the following page.

The configuration shown illustrates the use of the ServletRunner servlet engine used in the Solaris, HP-UX and AIX distributions. For Windows and Linux distributions, the configuration uses the Tomcat servlet engine. For Tomcat-specific port assignments, please see *DDE / Tomcat Port Changes* on page 245.

Port	Description	Names	Set In	Default Value
Translation request	QServer listens on this port for incoming translation requests.	REQUEST_PORT	<DDEInstallDir>\ qServerConfig.txt	7071
		serviceManagerRequestPort	<DDEInstallDir>\Jsdk\ webpages\WEB-INF\ servlets.properties	
FME service registration	QServer listens on this port for FME Servers wanting to register themselves.	SERVICE_REGISTRATION_PORT	<DDEInstallDir>\ qServerConfig.txt	7070
		Hardcoded in FME Server startup command	<DDEInstallDir>\ processMonitorConfig.txt	
Administration request	QServer listens on this port for incoming administration requests.	ADMINISTRATION_REQUEST_ PORT	<DDEInstallDir>\ qServerConfig.txt	7072
Servlet runner request	Servlet Runner listens on this port for incoming servlet requests.	server.port	<DDEInstallDir>\Jsdk\ default.cfg	8194
		servletURL	<DDEInstallDir>\Jsdk\ webpages\WEB-INF\ servlets.properties	
		Hardcoded in URLs	HTML files	

The figure below illustrates how the ports are set for each DDE component when the default ServletRunner servlet engine is used.

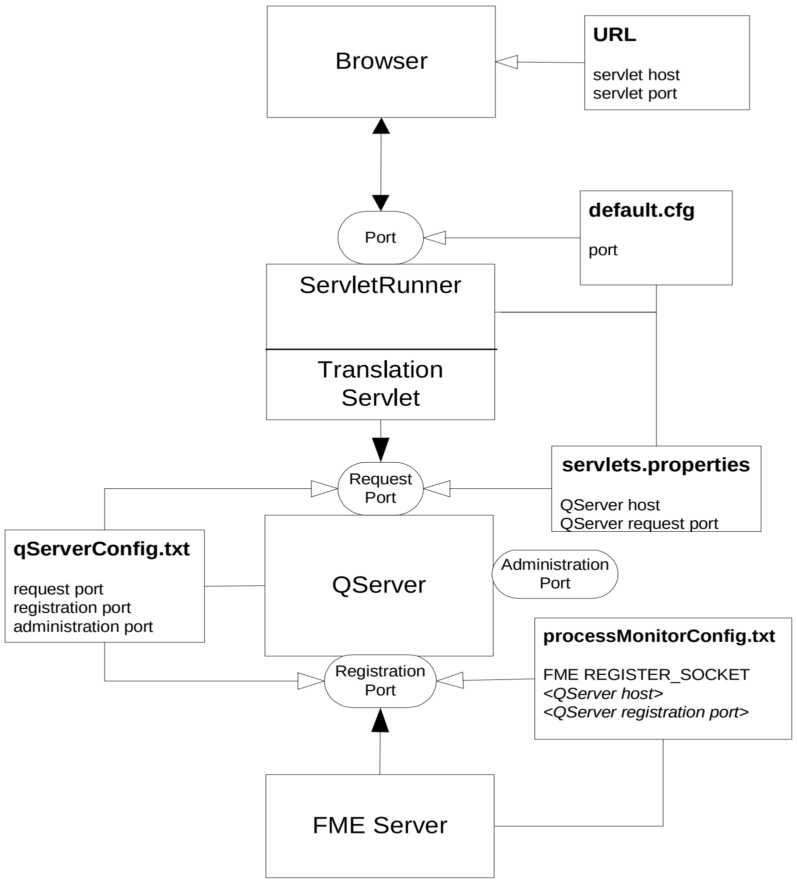


FIGURE C-1 **TCP/IP Port Settings**

APPENDIX

D

Status Return Codes

The QServer component returns status codes that can be used by programmers writing applications that communicate with DDE.

The status codes returned by this component is described below.

QServer API Status Return Codes

The methods of the QServerAPI class return the `kSPSuccess` integer value for successful operations and throw a `ServiceProviderException` if an error occurs.

The `ServiceProviderException` class is defined in `COM.safe.serviceprovider.ServiceProviderException`.

The exception message strings contained in these `ServiceProviderExceptions` have the following syntax:

```
<status code integer>:<status message>
```

An exception's status code and message values are determined by the specific condition that caused the exception and by the component that detected the condition. The condition can be detected either by the QServerAPI object that interfaces to the QServer or by the QServer itself. In either case, it is always the QServerAPI that throws the exception.

For example, if the QServerAPI object cannot connect to the QServer's request port, the condition is detected by the QServerAPI, and the thrown exception's message string is:

```
393602:SPRequestConnectFailed
```

If the QServer cannot obtain a translation result from the FME Server, the condition is detected by the QServer, and the thrown exception's message string is:

```
393407:SMGettingTransactionResultFailed
```

When an exception occurs, DDE's Translation Servlet displays the `ServiceProviderException` message in the web page it returns to the browser, and also enters the message into its log file.

Client programs can invoke `getMessage()` on the thrown `ServiceProviderException` and parse the returned message string using the colon delimiter to obtain the separate status code and message.

The QServerAPI and the QServer each define their own constants to represent status codes and messages for the conditions each detects. The values parsed from the exception message can be tested against the code constants defined in the interfaces described below.

QServerAPI Status Return Codes

The QServerAPI status return codes have the following characteristics:

Defined in package: `COM.safe.serviceprovider`
Defined by interface: `IServiceProviderConstants`
Message prefix: `SP`
Code number range: `393601-393700`

Code	Message	Description
0	SPSuccess	successful operation
393601	SPInitFailed	either couldn't get local host's IP address or couldn't create result server socket.
393602	SPRequestConnectFailed	couldn't connect to QServer's request port.
393603	SPRequestDisconnectFailed	couldn't disconnect from QServer's request port.
393604	SPResultConnectWaitFailed	couldn't wait for connect request by QServer to send translation result.
393605	SPBadTransactionRequest	translation request string created by QServer API and sent to QServer has invalid syntax.
393606	SPResultDisconnectFailed	couldn't disconnect from QServer result port.
393607	SPBadTransactionResult	translation result string returned by QServer has invalid syntax.
393608	SPRequestAcknowledgementFailed	couldn't get request acknowledgement response from QServer.
393609	SPResultDeliveryFailed	couldn't get translation result from QServer.
393610	SPTransactionStartFailed	couldn't send translation request string to QServer.
393611	SPIDSendingFailed	couldn't send ID to QServer after connecting to its request port.

The QServerAPI status return code constants are defined as follows:

QServerAPI Constant Definitions

```
package COM.safe.serviceprovider;
public interface IServiceProviderConstants
{// Constant prefix: kSP
// Message number range: 393601-393700.
// Status return values.
public static final int kSPSuccess                              = 0;
public static final String kSPSuccessMsg                        =
"SPSuccess";
public static final int kSPInitFailed                         = 393601;
public static final String kSPInitFailedMsg                     =
"SPInitFailed";
public static final int kSPRequestConnectFailed                 =
393602;
public static final String kSPRequestConnectFailedMsg           =
"SPRequestConnectFailed";
public static final int kSPRequestDisconnectFailed              =
393603;
public static final String kSPRequestDisconnectFailedMsg        =
"SPRequestDisconnectFailed";

public static final int kSPResultConnectWaitFailed              =
393604;
public static final String kSPResultConnectWaitFailedMsg        =
"SPResultConnectWaitFailed";
public static final int kSPBadTransactionRequest                =
393605;
public static final String kSPBadTransactionRequestMsg          =
"SPBadTransactionRequest";
public static final int kSPResultDisconnectFailed               =
393606;
public static final String kSPResultDisconnectFailedMsg         =
"SPResultDisconnectFailed";
public static final int kSPBadTransactionResult                 =
393607;
public static final String kSPBadTransactionResultMsg           =
"SPBadTransactionResult";
public static final int kSPRequestAcknowledgementFailed         =
393608;
public static final String kSPRequestAcknowledgementFailedMsg =
"SPRequestAcknowledgementFailed";
public static final int kSPResultDeliveryFailed                 =
393609;
public static final String kSPResultDeliveryFailedMsg           =
"SPResultDeliveryFailed";
public static final int kSPTransactionStartFailed               =
393610;
public static final String kSPTransactionStartFailedMsg         =
"SPTransactionStartFailed";
public static final int kSPIDSendingFailed                    = 393611;
public static final String kSPIDSendingFailedMsg                =
"SPIDSendingFailed";
```

QServer Status Return Codes

The QServer status return codes have the following characteristics:

```
Defined in package: COM.safe.servicemanager
Defined by interface: IServiceManagerConstants
Message prefix: SM
Code number range: 393401-393500
```

Code	Message	Description
0	SMSuccess	successful operation.
393401	SMRegistrationPortalInitFailed	couldn't initialize portal through which to accept FME Server registrations.
393402	SMTransactionPortalInitFailed	couldn't initialize portal through which to perform FME Server translations.
393403	SMRequestPortalInitFailed	couldn't initialize portal through which to accept translation requests.
393404	SMTransactionInitFailed	a Transaction object couldn't be initialized due to an invalid request or result message string (likely missing fields).
393405	SMInitFailed	QServer couldn't initialize itself due to missing config file or missing/bad parameter values in config file.
393406	SMTransactionStartFailed	couldn't get translation request sent to/started by FME Server.
393407	SMGettingTransactionResultFailed	couldn't get translation result from FME Server.
393408	SMSendingAdminTransactionResultFailed	couldn't send admin result to client.

The QServer status return code constants are defined as follows:

QServer Constant Definitions

```
package COM.safe.servicemanager;
public interface IServiceManagerConstants
{
// Constant prefix: kSM
// Message number range: 393401-393500.
// Status return values.
public static final int kSMSuccess                            = 0;
public static final String kSMSuccessMsg                      =
"SMSuccess";
public static final int kSMRegistrationPortalInitFailed = 393401;
public static final String kSMRegistrationPortalInitFailedMsg   =
"SMRegistrationPortalInitFailed";
public static final int kSMTransactionPortalInitFailed = 393402;
public static final String kSMTransactionPortalInitFailedMsg    =
"SMTransactionPortalInitFailed";
public static final int kSMRequestPortalInitFailed = 393403;
public static final String kSMRequestPortalInitFailedMsg        =
"SMRequestPortalInitFailed";
public static final int kSMTransactionInitFailed = 393404;
public static final String kSMTransactionInitFailedMsg          =
"SMTransactionInitFailed";
public static final int kSMInitFailed                           =
393405;
public static final String kSMInitFailedMsg                     =
"SMInitFailed";
public static final int kSMTransactionStartFailed = 393406;
public static final String kSMTransactionStartFailedMsg         =
"SMTransactionStartFailed";
public static final int kSMGettingTransactionResultFailed = 393407;
public static final String kSMGettingTransactionResultFailedMsg =
"SMGettingTransactionResultFailed";
public static final int kSMSendingAdminTransactionResultFailed =
393408;
public static final String kSMSendingAdminTransactionResultFailedMsg
= "SMSendingAdminTransactionResultFailed";
```

APPENDIX

E

Disabling the DDE Port Forwarding Feature

This appendix assumes that DDE has been installed in its default configuration with the Translation Servlet being run by either Tomcat (Windows and Linux installations) or ServletRunner (Solaris, HP-UX and AIX installations) and communicating on port 8194.

In this configuration, a CGI Perl script is utilised to forward requests from a client application (usually a web browser) communicating with the web server on port 80 to the Translation Servlet via the servlet engine, communicating on port 8194.

Why Use Port Forwarding?

Port forwarding has been utilized to help in situations where a firewall has been implemented on the web server machine and communications, other than through the standard port 80, have been restricted. In this case, it may be impossible for a client application to communicate directly with the Translation Servlet on port 8194 and your network administrator may be reluctant to open this non-standard port for communications.

Port forwarding, as implemented by DDE, is extremely lightweight, simple to use and has no significant impact on DDE or web server performance. Therefore, it is installed as the default behavior for DDE even in situations where no firewall or other communications restrictions exist.

An Alternative to Disabling Port Forwarding

Even in instances where a DDE installation has been customized so that communications to the Translation Servlet no longer occur on port 8194, it is usually more simple to adjust the Perl port forwarding script to forward requests to the new location rather than completely disabling the port forwarding mechanism.

To adjust the port forwarding location:

1 Open the file `<web_server_root>/<cgi-bin>/DDE/spatialDirect.pl` in a text editor for editing.

  For example, for a default Apache web server, installed on Windows, you would open the file:

  ```
  C:\Program Files\Apache Group\Apache2\cgi-bin\DDE\
  spatialDirect.pl
  ```

  For a default Microsoft IIS web server installation, the file would be:

  ```
  C:\Inetpub\Scripts\DDE\spatialDirect.pl
  ```

2 Search for the line that has the form:

  ```
  my $forwardingURL = URI->new('http://
  <web_server_name>:<servletPort>/servlet/
  translationServlet');
  ```

  For example:

  ```
  my $forwardingURL = URI->new('http://rohm:8194/servlet/
  translationServlet');
  ```

3 Replace the web server name and port with your new details.

Note: The web server name will likely stay the same; however, the port will probably change.

For example, to change the above example to forward requests to port 8000, rather than port 8194, the new line would be:

```
my $forwardingURL = URI->new('http://rohm:8000/servlet/
translationServlet');
```

4 Save your changes to spatialDirect.pl and attempt a new download translation. (There is no need to restart DDE or your web server.)

Note: If port forwarding is not required (that is, if communications with the Translation Servlet are to occur on the standard web server port of 80), it is still possible to use the port forwarding mechanism and simply replace the servlet port (8194) with the default port of 80. You can even to remove the :port number reference entirely from the forwarding address to use 80 by default. For example:

```
   my $forwardingURL = URI->new('http://rohm/servlet/
   translationServlet');
```

Disabling Port Forwarding

Why disable port forwarding? In a default DDE installation there is very little reason to disable port forwarding except, perhaps, if CGI scripting has been disabled for your web server as a security measure. However, if your DDE installation has been modified to make use of a custom servlet engine which communicates on a port other than port 8194, or if your web server contains an integrated servlet engine and can handle servlet communications directly through port 80 then disabling port forwarding may be an option to consider.

To disable the CGI Perl script port forwarding mechanism:

1 Stop DDE.
For detailed instructions on stopping DDE, please refer to the appropriate section of this manual:
   - *Stopping DDE on Windows* on page 42
   - *Stopping DDE on UNIX* on page 52
2 Update the Translation Servlet properties.
These instructions assume that a default DDE installation is running and that the Translation Servlet is being run by either Tomcat or ServletRunner, depending on your operating system. If a custom servlet engine is being used to run the Translation Servlet then you will need to identify the appropriate file or method used for modifying the servlet properties for your servlet engine and update the property described below.

   Open the translation servlet properties file for editing. This file will be in a different location and format depending on the operating system on which DDE has been installed.

   Windows and Linux:

```
<DDEInstallDir>/tomcat/webapps/servlet/WEB-INF/web.xml
```

   Solaris, HP-UX and AIX:

```
<DDEInstallDir>/Jsdk/webpages/WEB-INF/servlets.properties
```

   Search for and replace the definition of the servletURL property to call the Translation Servlet directly on the desired port, rather than through the port forwarding Perl script. For example, to call the Translation Servlet directly on port 8194 modify the servletURL property as follows:

   On Windows and Linux, change:

```
<init-param> <param-name>servletURL</param-name>  <param-
value>http://<your_web_server>:80/cgi-bin/DDE/
spatialDirect.pl</param-value> </init-param>
```

   to:

```
<init-param> <param-name>servletURL</param-name>  <param-
value>http://<your_web_server>:8194/servlet/
translationServlet</param-value> </init-param>
```

   On Solaris, HP-UX and AIX, change:

```
servletURL=http://<your_web_server>:80/cgi-bin/DDE/
spatialDirect.pl,\
```

to:

```
servletURL=http://<your_web_server>:8194/servlet/
translationServlet,\
```

3 Update affected web pages.

Navigate to the directory `<DDEInstallDir>/ArcIMS/safeViewerHTML`

Edit *all* HTML files in this directory and replace *all* references to:

```
http://<your_web_server>:80/<cgi-bin>/DDE/spatialDirect.pl
```

with:

```
http://<your_web_server>:<port>/servlet/translationServlet
```

For example, an HTML file for an installation running Microsoft IIS web server might contain a reference like:

```
http://rohm:80/Scripts/SpatialDirect/
spatialDirect.pl?SSFunction=prepareFetch
```

which, when changed to call the Translation Servlet directly on port 8194, would become:

```
http://rohm:8194/servlet/
translationServlet?SSFunction=prepareFetch
```

4 Start DDE.

For detailed instructions on starting DDE, please refer to the appropriate section of this manual:

- *Starting DDE on Windows* on page 40
- *Starting DDE on UNIX* on page 48

Note: If a custom servlet engine is being used, then it may also be necessary to stop and restart your servlet engine and/or web server for the completed updates to take effect.

APPENDIX F

Troubleshooting

DDE problems often involve startup failures of one or more of its components. Once all components have started and are running successfully, any subsequent errors usually involve data-specific translation issues encountered by the FME Server itself.

The following sections describe general troubleshooting for DDE.

Log and Configuration Files

Each of the four DDE components – Process Monitor, Translation Servlet, QServer and FME Server – has an associated log file and configuration file. Both are useful tools in diagnosing DDE problems. The names and locations of DDE log and configuration files are given in *What to Send* on page 302.

In general, whenever a particular DDE component is suspected of having a problem, its log file is first examined for relevant error messages. The absence of a component's log file is also useful information, since it indicates that the component failed to start up at all.

After examining the log file, it can also be useful to examine the component's configuration file. Inappropriate configuration settings can cause a component to fail, either on its own or in conjunction with one or more of the other DDE components.

The following diagram shows the DDE components in a typical configuration, along with the log files for each.

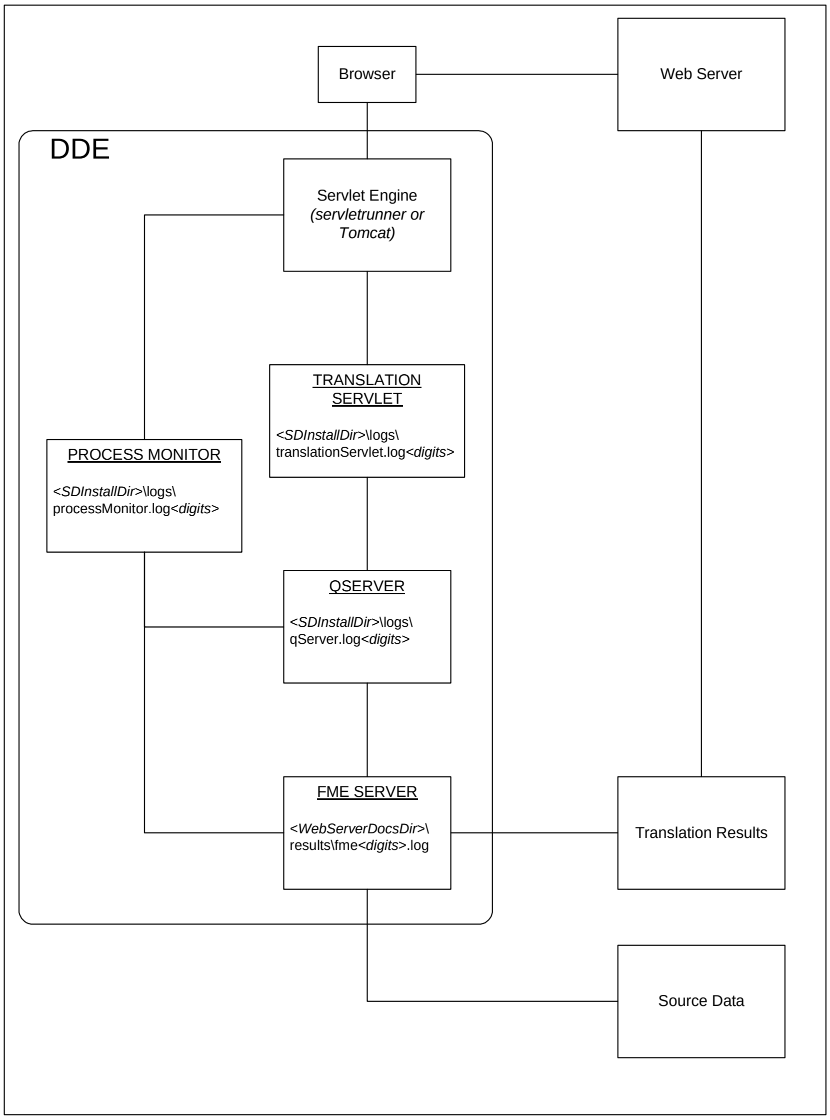

Browser
Web Server
DDE
Servlet Engine *(servletrunner or Tomcat)*
TRANSLATION SERVLET
<*SDInstallDir*>\logs\translationServlet.log<*digits*>
PROCESS MONITOR
<*SDInstallDir*>\logs\processMonitor.log<*digits*>
QSERVER
<*SDInstallDir*>\logs\qServer.log<*digits*>
FME SERVER
<*WebServerDocsDir*>\results\fme<*digits*>.log
Translation Results
Source Data

FIGURE F-1 DDE **Components and Associated Log Files**

Browser Display Sequence Errors

A common area where DDE problems first become apparent is in the browser while attempting to log in and make a translation request. A successful request generates a typical sequence of browser display pages. The failure of one of these pages to appear in the sequence can be used to determine which DDE components are operating correctly and which one may be causing the failure.

The following table shows the commonly encountered browser display sequence, what the presence and absence of each page implies, and what general actions to take if one is absent.

Does browser display...	If YES then...	If NO then...
Order Form/Download page	Translation Servlet OK	• Process Monitor was not started. • Servlet engine (ServletRunner or Tomcat) not running. • Translation Servlet not runnable. • Check if Process Monitor log file exists, and if so, check it to see if servlet engine started. • Check if Translation Servlet log file exists and if so, check it for error messages.
translation result page (image map display or zip download)	QServer and FME Server OK	• Process Monitor never started. • QServer and/or FME Server not running. • FME Server not licensed for DDE. • Check if Process Monitor log file exists, and if so, check it to see if QServer and FME Server started and if FME Server is licensed. • Check if QServer log file exists and if so, check it for error messages.
successful translation message	Source Data OK; Translation OK; Result Location OK	• FME can’t read/understand source data. • FME can’t translate between specified formats. • FME can’t write to result location. • Check FME Server translation log for error messages.

Stopping DDE on Windows

If an attempt is made to erroneously stop an interactively started DDE instance by using a mechanism other than the Windows **Start | All Programs | DDE | Stop DDE** command, it may be necessary to clean up orphaned processes before it is possible to start DDE again. This may be done by either:

- Rebooting the DDE server computer, or
- opening the Windows Task Manager and killing orphaned processes.

Terminating Orphaned Process Using Windows Task Manager

1. Start Windows Task Manager. Choose Start | Run, type “taskmgr” and press Enter
2. Select the "Processes" tab in the Task Manager window
3. Click the "Image Name" heading to sort the processes by process name
4. Identify any of the following processes which are still running:
   - `fme.exe`
   - `java.exe (3 instances)`

   Warning

   Other applications may be running processes named `java.exe`. If in doubt, do NOT terminate these processes manually; instead, reboot your computer.

5. Terminate the above named processes by selecting each one in turn and clicking the "End Process" button. When prompted for confirmation to terminate the process, click "Yes".

Usage Issues

Internet Explorer Timeout During Translation

The Microsoft Internet Explorer browser has a default time-out of 5 minutes, meaning that if it receives no response within 5 minutes of its request, the browser will assume that no response is forthcoming and will terminate the connection to the web server.

If Internet Explorer sends a DDE translation request, and the translation takes longer than 5 minutes to complete, the browser will terminate the connection without waiting to receive the results. The results will have been created, but they will not be deliverable, since the browser will no longer be listening for them.

This can be prevented by increasing Internet Explorer's timeout value. This requires ensuring that a minimum version of Internet Explorer is used and that an entry is present in the NT system registry. To increase the timeout value, perform the following steps:

1. Ensure that Internet Explorer Version 4.01 Service Pack 1 or later is installed. To obtain this, go to the following website:
   ```
   http://www.microsoft.com/ie/
   ```
   and download and install the required version.
2. Make the required system registry entry as follows:
   Click on Start | Run..., enter `regedit` in the Run window that appears, then click OK.
3. In the Registry Editor window, select the following registry key:
   ```
   HKEY_CURRENT_USER\Software\Microsoft\Windows\
   CurrentVersion\Internet Settings
   ```
4. Click on Edit | New | DWORD Value.
5. Rename the newly created entry from its temporary name, such as `New Value #1`, to `ReceiveTimeout`.
6. With the newly created `ReceiveTimeout` entry still selected, click on Edit | Modify.
7. In the Edit DWORD Value window, select the Decimal choice in the Base box, and enter the desired timeout value in the Value data: entry field. This value must be the number of milliseconds that the browser will wait before timing out. It should be made at least as long as the longest translation time anticipated.
   For instance, if 1 hour is the desired time-out value, enter the value (1 * 3600) * 1000 = 3600000. Then click OK and exit the Registry Editor.

8 Reboot the computer for the new value to take effect. The Internet Explorer browser should now time-out after the new, longer duration, allowing results from long translations to be delivered.

Using IIS Web Server on Non-Server Windows 2000

When DDE is used in conjunction with the IIS web server on a *non-server* version of Windows 2000, problems may occur when response pages are sent back to the browser. These problems include display of incomplete pages (missing graphics, text, etc) and error messages indicating that pages cannot be found. These errors occur inconsistently and with variable frequency.

Note that this problem occurs only when IIS is used on a non-server version of Windows 2000. When the server version is used, the problem does not occur.

This problem is caused by a temporary lack of available connections within IIS. The non-server Windows 2000 version limits the maximum number of simultaneous IIS connections to 10. (The server version allows a greater number of simultaneous IIS connections).

The problem can be greatly reduced by disabling the use of keep-alive connections by IIS. This is done by navigating to the IIS manager tool as follows:

Start > Control Panel > Administrative Tools > Internet Services Manager

Then right-click on the default website entry and select Properties. On the resulting properties panel website tab, examine the Connections control group and ensure that the `HTTP Keep-Alives Enabled` checkbox is *un*checked, thereby disabling keep-alive connections.

Sending Result Notification E-mail

In DDE V1.3.1 and later, result notification e-mail is sent using the platform independent JavaMail API. However, in DDE versions prior to V1.3.1, platform-specific methods are used to send e-mail. This section describes the older method used on Windows NT systems.

NT Notification E-mail Prior to V1.3.1

On DDE installations prior to V1.3.1, e-mail on NT systems is sent by spawning a process and running a public domain command line mailing program called `blat`, developed by P. Mendes, M. Neal, G. Vollant and T. Charron. Blat is located in:

```
<DDEInstallDir>\util\blat.exe
```

The SMTP server used by `blat` is set during `blat`'s installation, which occurs during DDE installation. As a result, the SMTP server parameter is actually part of `blat`'s configuration, not DDE's.

`Blat` stores its config info (including the SMTP server name) in the NT system registry when `blat` is installed. When `blat` sends e-mail, it determines the SMTP server from the registry. System crashes and power failures can sometimes corrupt registry settings, including those used by `blat`, and this can prevent `blat` from connecting to the mail server.

`Blat` system registry settings are stored in registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Public Domain\Blat
```

The two data names of greatest relevance are:

```
SMTP server
Sender
```

The values of these names can be manually specified using the following blat installation command (this command is also the one executed by the DDE installation procedure):

```
<DDEInstallDir>\util\blat.exe -install <SMTPHostName>
<SMTPSenderAddress>
```

If `blat` registry settings are incorrect or corrupted, blat can be reinstalled manually using the above command, which should reset the registry to correct blat values. The `<SMTPSenderAddress>` is an e-mail address on the SMTP host that blat uses as the sender of the notification e-mail.

On-line `blat` help can be obtained by running the following:

```
<DDEInstallDir>\util\blat.exe -h
```

E-mail text file contents can be manually sent for testing purposes by running the following:

```
<DDEInstallDir>\util\blat.exe <filename> -to <recipient>
```

Firewall Configuration

Additional configuration issues can arise when installing DDE behind firewall security systems.

Consider a typical firewall-based system configuration shown in the figure below. DDE and a web server are running on `Neptune` which is accessible to external users via a firewall running on `Zeus,` using the URL `www.company.com`.

External users should be able to access DDE and the results it generates while being able to see `Zeus` but not `Neptune`. This is accomplished by the following:

- The firewall is set to map incoming URL requests to `Neptune`. So, for example, the URL `www.company.com` is sent to default port 80 on `Neptune`, where the web server is listening.
- Network parameters on `Neptune` are set to map the `www.company.com` URL to `Neptune's` own IP address. This allows the Translation Servlet component of DDE to obtain the internally generated files needed to display legends and image maps in its GIF output.
- When the DDE installation procedure asks for the IP address of the QServer system, the address of `Neptune` is given, not `Zeus`. In general, the QServer system specified within DDE must be the internal system on which the QServer is actually running. This is necessary to allow the Translation Servlet and the FME Server to connect to the QServer.
- When the DDE installation procedure asks for the network name of a host system for result notification e-mail, the name of an SMTP e-mail system should be given. The Translation Servlet system (`Neptune` in this case) must be able to see this mail host system, and the mail host system must be able to send e-mail to the external users.
- When the DDE installation procedure asks for the network names of any other systems, the URL presented to external users by the firewall system should be given (`www.company.com` in this case). This allows users to access and download translation results via the website address without seeing the internal network protected by the firewall.

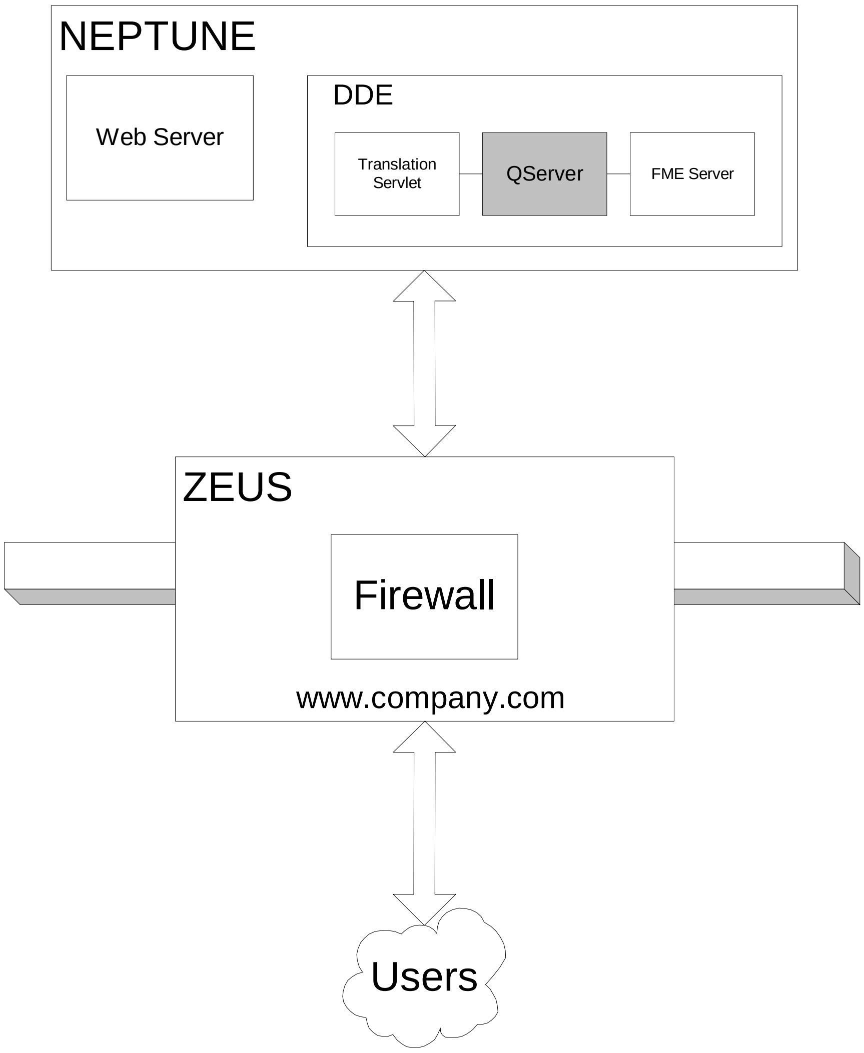


FIGURE F-2 **Example Firewall Configuration**

Where to Obtain Help

If you need help with DDE, please contact your local ESRI representative.

What to Send

To assist them in tracking down problems with your DDE installation, please send the most recent copies of as many of the files listed below as possible. The names reflect the standard default DDE configuration. The `<digits>` in the log file names may or may not be present.

It's usually best to e-mail all of these files as a zip-compressed attachment, along with a description of the problem and of any other relevant site-specific factors.

Also, if at all possible, please check the browser display sequence described above, and use the table to try to determine which component(s) may be failing. Please include any results of this check in your e-mail along with the attached files.

Log Files to Send

Process Monitor

```
<DDEInstallDir>\logs\processMonitor.log<digits>
```

Translation Servlet

```
<DDEInstallDir>\logs\translationServlet.log<digits>
```

QServer

```
<DDEInstallDir>\logs\qServer.log<digits>
```

FME Server

```
<WebServerDocsDir>\<SDResultsDir>\fme<digits>.log
```

Configuration Files to Send

Process Monitor

```
<DDEInstallDir>\processMonitorConfig.txt
```

Translation Servlet Servlet Engine

For Windows and Linux systems:

```
<DDEInstallDir>\tomcat\webapps\servlet\WEB-INF\web.xml
```

For all other systems:

```
<DDEInstallDir>/Jsdk/webpages/WEB-INF/servlets.properties
```

QServer

```
<DDEInstallDir>\qServerConfig.txt
```

FME Server

```
<DDEInstallDir>\fmeServerConfig.txt
<DDEInstallDir>\translationControl\system\config.csv
```